2

NAVAL POSTGRADUATE SCHOOL
Monterey, California

UNITED STATES NAVY
NAVAL POSTGRADUATE SCHOOL

THESIS

DATA ADMINISTRATION FOR THE RAPID ACQUISITION OF MANUFACTURED PARTS

by

Catherine T. Eads

and

Pamela A. Smith

September 1988

Thesis Advisor: Daniel R. Dolk

UNCLASSIFIED
SECURITY CLASSIFICATION OF THIS PAGE

ADA201859

REPORT DOCUMENTATION PAGE

1a. REPORT SECURITY CLASSIFICATION UNCLASSIFIED	1b RESTRICTIVE MARKINGS
2a. SECURITY CLASSIFICATION AUTHORITY	3. DISTRIBUTION/AVAILABILITY OF REPORT Approved for public release; distribution is unlimited
2b. DECLASSIFICATION/DOWNGRADING SCHEDULE	
4. PERFORMING ORGANIZATION REPORT NUMBER(S)	5. MONITORING ORGANIZATION REPORT NUMBER(S)
6a. NAME OF PERFORMING ORGANIZATION Naval Postgraduate School; 6b OFFICE SYMBOL (If applicable) Code 54	7a. NAME OF MONITORING ORGANIZATION Naval Postgraduate School
6c. ADDRESS (City, State, and ZIP Code) Monterey, California 93943-5000	7b. ADDRESS (City, State, and ZIP Code) Monterey, California 93943-5000
8a. NAME OF FUNDING/SPONSORING ORGANIZATION; 8b. OFFICE SYMBOL (If applicable)	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER
8c. ADDRESS (City, State, and ZIP Code)	10. SOURCE OF FUNDING NUMBERS: PROGRAM ELEMENT NO. / PROJECT NO. / TASK NO / WORK UNIT ACCESSION NO

11. TITLE (Include Security Classification)
DATA ADMINISTRATION FOR THE RAPID ACQUISITION OF MANUFACTURED PARTS

12. PERSONAL AUTHOR(S)
Eads, Catherine T. and Smith, Pamela A.

13a. TYPE OF REPORT	13b TIME COVERED	14 DATE OF REPORT (Year, Month, Day)	15 PAGE COUNT
Master's Thesis	FROM ____ TO ____	1988, September	106

16. SUPPLEMENTARY NOTATION
The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.

17 COSATI CODES			18 SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
FIELD	GROUP	SUB-GROUP	Data Administration; Flexible Manufacturing System

19 ABSTRACT (Continue on reverse if necessary and identify by block number)

Procurement of spare parts is both time consuming and costly for the Navy. The Rapid Acquisition of Manufactured Parts (RAMP) is a Navy program designed to reduce the lead time required to procure small mechanical parts by up to 90%. RAMP is a flexible manufacturing system (FMS) which will use computer controlled equipment to produce 15,000 parts per year with an average lot size of four parts. A distributed system consisting of heterogeneous hardware, software and data, the RAMP environment presents many database administration difficulties.

This thesis presents an overview of the RAMP Manufacturing System, discusses the data administration issues found in distributed computing environments and suggests an expanded information resource dictionary system to manage and control RAMP's shared data. The

20 DISTRIBUTION/AVAILABILITY OF ABSTRACT [✓ UNCLASSIFIED/UNLIMITED] ☐ SAME AS RPT ☐ DTIC USERS	21 ABSTRACT SECURITY CLASSIFICATION Unclassified
22a NAME OF RESPONSIBLE INDIVIDUAL Prof. Daniel R. Dolk	22b TELEPHONE (Include Area Code) (408) 646-2260; 22c OFFICE SYMBOL Code 54Dk

DD FORM 1473, 84 MAR   83 APR edition may be used until exhausted. All other editions are obsolete   SECURITY CLASSIFICATION OF THIS PAGE

U.S. Government Printing Office: 1986—606-2

Best Available Copy

UNCLASSIFIED

#19 - ABSTRACT - (CONTINUED)

problem of maintaining consistency among multiple databases in the event of a failure is examined.

Data Administration for the Rapid Acquisition of Manufactured Parts

by

Catherine T. Eads
Lieutenant, United States Navy
B.A., Southern Illinois University, 1978

and

Pamela A. Smith
Lieutenant, United States Naval Reserve
B.S., Park College, 1981

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN INFORMATION SYSTEMS

from the

NAVAL POSTGRADUATE SCHOOL
September 1988

Authors: [illegible] [illegible]
Catherine T. Eads   Pamela A. Smith

Approved by: [illegible]
Daniel R. Dolk, Thesis Advisor

[illegible]
Alan W. McMasters, Second Reader

[illegible]
David R. Whipple, Chairman
Department of Administrative Sciences

[illegible]
Kneale T. Marshall
Dean of Information and Policy Sciences

ABSTRACT

Procurement of spare parts is both time consuming and costly for the Navy. The Rapid Acquisition of Manufactured Parts (RAMP) is a Navy program designed to reduce the lead time required to procure small mechanical parts by up to 90%. RAMP is a flexible manufacturing system ~~(FMS)~~ which will use computer controlled equipment to produce 15,000 parts per year with an average lot size of four parts. A distributed system consisting of heterogeneous hardware, software and data, the RAMP environment presents many database administration difficulties.

This thesis presents an overview of the RAMP Manufacturing System, discusses the data base administration issues found in distributed computing environments and flexible manufacturing systems, and suggests an expanded information resource dictionary system to manage and control RAMP's shared data. The problem of maintaining consistency among multiple databases in the event of a failure is examined.

TABLE OF CONTENTS

I. INTRODUCTION

A. BACKGROUND

Computers have revolutionized manufacturing production processes. "Advances in automation technology will soon yield fully automated factories which operate under the concept of computer integrated manufacturing (CIM)." [Ref. 1:p. 8] Beginning with an order request, CIM automation will eventually direct all stages of the production process from initial order receipt to design, engineering, production and actual shipment. While CIM has yet to be fully realized, a companion technology and prerequisite to complete factory automation does exist. Flexible Manufacturing Systems (FMS) are a way station towards CIM and are providing substantial benefits in manufacturing industries today.

Flexible manufacturing systems are "computer controlled production systems that produce a family of parts in a flexible manner." [Ref. 2:p. 11] Parts that are similar in size and shape, or which must flow through a similar production process are grouped into families. Software then dictates the actual part to be produced, thus eliminating the expensive and time-consuming process of retooling that is common to traditional manufacturing environments. An additional benefit of FMS is that it dramatically reduces lead time requirements.

The U.S. Navy recognizes these benefits, and has developed a test and integration facility at Charleston Naval Shipyard (CNSY) for the development of an FMS. The program--Rapid Acquisition of Manufactured Parts (RAMP)--is estimated to reduce lead-time requirements up to 90%, thereby providing a substantial benefit to fleet operational capabilities. While traditional manufacturing is geared towards the mass production of a single item, RAMP is geared toward the production of any one of a family of 15,000 parts with an average lot size of only four. Once operational, the concept will be installed at CNSY and other naval shipyards, and eventually to private industries if demand warrants.

RAMP is an hierarchical distributed system containing heterogeneous hardware, software and data. It includes various numerical control machine tools which perform the physical manipulations required to produce any one of the cylindrical or prismatic parts to be manufactured. A three-level hierarchy--cell, workstation, and device--is employed to control processing. Each item produced goes through four functional components--Manufacturing Engineering, Production and Inventory Control, Manufacturing, and Quality--before it is shipped to its destination. All components are integrated, and generally require information from previous steps to perform their tasks. A fifth function--Information Management and Communication--provides the linking mechanism between these components.

This kind of computerized manufacturing environment presents many difficulties when it comes to designing an appropriate architecture. This is especially true with respect to the various databases which will exist in the system. Heterogeneous distribution requires more careful consideration of certain data administration issues than do centralized, or even homogeneous distributed systems. Communication is complicated by the need to translate different "languages" used by the various functional components, and to transfer and share data between components. A network also requires additional considerations for security. Simultaneous access requirements by different components of the system require mechanisms to prevent the database situations known as deadlock and livelock, and to ensure that one update does not overwrite another. If two processes are simultaneously waiting for access to data that the other controls, neither process can proceed and deadlock results. If, on the other hand, a priority system were employed to determine which process proceeds first, mechanisms must be in place to ensure that the other process is not continually preempted thus causing livelock.

Another critical issue in a distributed FMS environment is determining the scope or granularity of a transaction which dictates exactly when various databases are physically modified. Does a transaction encompass the entire production of a part, or can transactions be decomposed into smaller processes such as cutting or boring a piece of metal? The strategy selected has significant implications for data integrity and database efficiency. In an atmosphere where accurate data is central to the very existence and efficient operation of the facility, backup and recovery functions are also exceedingly crucial. Finally, because there are many different components which require access to shared data, a system-wide information resource management mechanism must exist to coordinate these data-oriented activities and enforce the necessary degree of data administration.

The Integrated Manufacturing Database Administration System (IMDAS) was developed by the National Bureau of Standards (NBS) as a baseline architecture for FMS. It provides a structure that can be tailored to meet the specific needs of a particular environment. While IMDAS does address the data administration issues which are generic to FMS, it cannot address those which will be peculiar to "tailored" implementations. RAMP therefore must consider the additional data administration issues that will be encountered in its unique environment. This issue is the primary focus of this thesis.

B. OBJECTIVE

The purpose of this thesis is to identify those aspects of data administration which we consider salient to RAMP that have not yet been adquately addressed in the documentation we've studied. In addition, we propose an information resource dictionary system (IRDS) to facilitate the data administration functions of the unique RAMP environment.

C. SCOPE

Much of the information upon which this thesis is based emanated from the RAMP Small Mechanical Parts Critical Design Review (SMP CDR) held in Charleston, SC August 30 and 31, 1988. However, neither the CDR nor any other document available to us during the preparation of this thesis provided any detailed specification of the RAMP database architecture. It is possible, therefore, that some of the observations and recommendations made herein may subsequently be inappropriate for the eventual RAMP database configuration. It is not our intention to criticize any subcontractors associated with the RAMP project but rather to shed light on database issues we think have been inadequately addressed from the limited information we had available.

D. PREVIEW

Chapter II provides an overview of the RAMP facility at CNSY. In Chapter III, the reader is given a general overview of distributed computer systems, and the data administration complexities found in heterogeneous CIM and FMS environments. Chapter IV proposes an Information Resource Dictionary System (IRDS) tailored to the unique needs of RAMP, and then addresses the specific data administration aspects which need to be more fully addressed in the RAMP architecture. Conclusions and recommendations are provided in Chapter V.

II. RAMP AND THE RAMP DATABASE

A. INTRODUCTION

The Rapid Acquisition of Manufactured Parts (RAMP) is a Navy program for the procurement of small mechanical parts (SMP) and printed wiring assemblies (PWA). This thesis, however, will focus only on the procurement of small mechanical parts. Procurement of spare parts is both time consuming and costly for the Navy. Small mechanical parts may take as long as 300-400 days to procure. The cost in terms of fleet readiness of such a delay is very high.

The procurement process begins when a ship or aircraft requisitions a part from the Ships Parts Control Center (SPCC) or the Aviation Supply Office (ASO). If the part is not available from the supply system, these inventory control points (ICPs) will issue an invitation for bid (IFB). Various manufacturers will then submit bid proposals to provide the part. Following the closing of the bidding process, the proposals are reviewed, a manufacturer is selected, and finally the contract is awarded. When drawings and specifications for the part are not readily available, the procurement process is delayed even further. The selected manufacturer begins the engineering and design process necessary to produce the part. The manufacturer must set up and retool for production whether the quantity is six or 6000. Eventually production is completed and parts are shipped. [Ref. 3:p. 4]

A ship requiring a particular part in order to deploy must obtain the part from another ship if it is not immediately available through the supply system. That ship in turn may need to obtain the part from still another ship in order to meet its deployment schedule. This process could be repeated several times before the part is received from a manufacturer 300-400 days later. During this period, fleet readiness has been impaired.

While maintaining large inventories is costly and inefficient, the long lead time required for procurement of replacement parts or spares is also unacceptable. Typically, procurement administrative lead time and manufacturing administrative lead time consume about 90% of the procurement cycle while actual production comprises about 10% [Ref. 4:p. 4]. The goals of RAMP are to reduce these long lead times, decrease costs, and improve fleet readiness by introducing technologies such as electronic data transfer, computer-aided design (CAD) and computer-aided manufacturing (CAM) into the Navy logistics system. As shown in Figure 1, procurement administrative lead time and manufacturing administrative lead time are expected to be reduced by up to 90%.

A RAMP Test and Integration Facility (RTIF) has been established in Charleston, South Carolina for the planning,

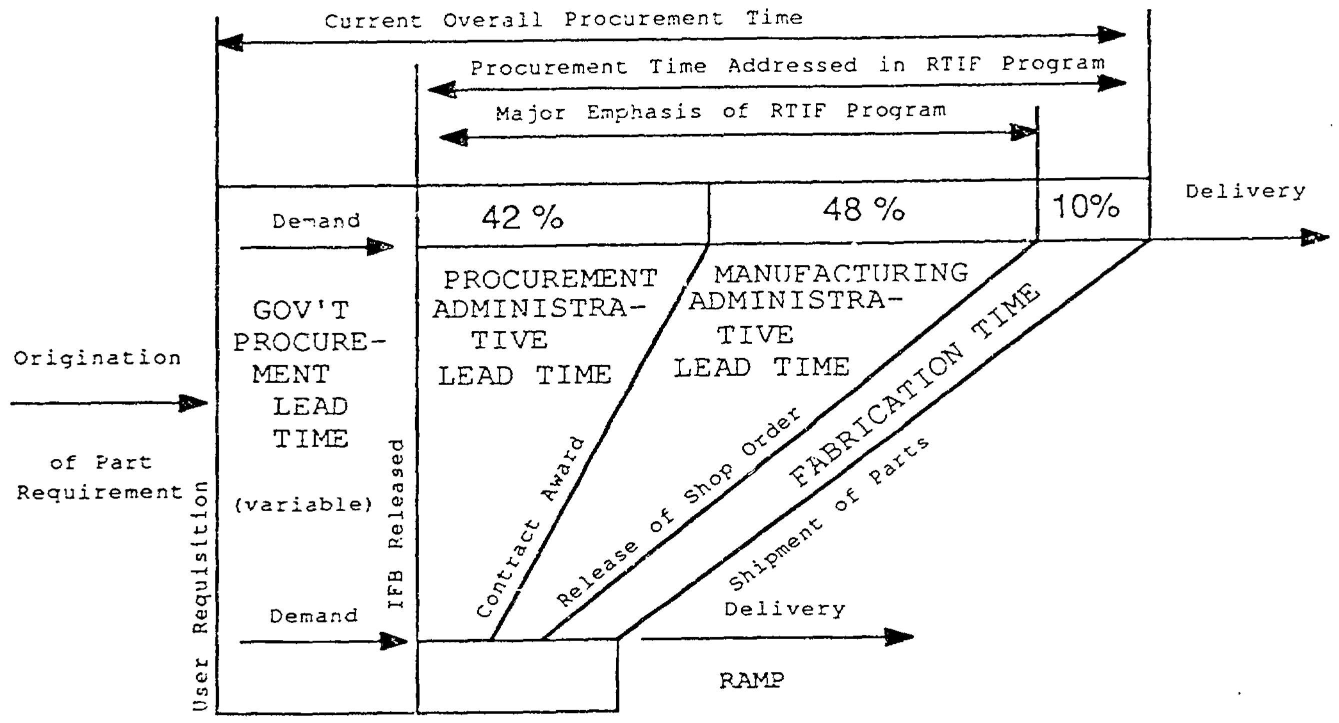

Current Overall Procurement Time
Procurement Time Addressed in RTIF Program
Major Emphasis of RTIF Program
Delivery
42 %
48 %
10%
Demand
Origination
of Part Requirement
User Requisition
GOV'T PROCURE- MENT LEAD TIME
(variable)
IFB Released
PROCUREMENT ADMINISTRA- TIVE LEAD TIME
MANUFACTURING ADMINISTRA- TIVE LEAD TIME
FABRICATION TIME
Contract Award
Release of Shop Order
Shipment of Parts
Demand
Delivery
RAMP

Figure 1. RAMP Reduces Lead Time [Ref. 4:p. 4]

development, testing, and demonstration of the RAMP concept. Once the RTIF becomes operational, a RAMP capability will be installed at the Charleston Naval Shipyard (CNSY). Transferring RAMP technology to other Naval facilities and to private industry is a secondary long term goal of the RAMP program. [Ref. 5:p. 6]

B. SYSTEM OVERVIEW

Traditional manufacturing is geared to the high volume production of a single item and does not lend itself to the rapid production of small quantities of many different parts. However, only one or a few parts may be needed. A Flexible Manufacturing System (FMS) is one in which it is possible to produce on demand any part from a selected group or family of parts, without incurring system downtime to change equipment [Ref. 6:p. 342]. Parts which have nearly the same size and shape, or parts which share the same or similar sequence of production operations are considered a family of parts.

1. Parts Produced by RAMP

The RAMP SMP Manufacturing System is designed to be an FMS and produce a wide variety of cylindrical and prismatic parts. Table 1 lists sample part types to be manufactured by the RAMP SMP system. Cylindrical parts will be not more than 24 inches in length and 12 inches in diameter. Prismatic parts will be not more than 24 inches in length, 8 inches in depth and 12 inches in height. Part

TABLE 1

TYPICAL TYPES OF SMALL MECHANICAL PARTS
TO BE PRODUCED BY THE RAMP SMP

Adapter	Fitting	Round Blank
Angle, leg	Gear	Round Flange
Ball	Gland	Round Pad
Bearing	Guide	Shaft
Body	Handle	Sleeve
Bolt	Nipple	Socket
Bonnet	Nozzle	Socket end
Boss	Nut	Spacer
Bracket	Pin	Stem
Bushing	Plate, backing	Stud
Butt, end	Plug	Support
Cam	Prismatic Blank	Tailpiece
Cap	Prismatic Flange	Tee
Case, gear	Prismatic Pad	Threadpiece
Collar	Reducer	Union
Connector	Ring	Washer
Coupling	Roller	"Y" Branch
Elbow		

Source: [Ref. 7:p. 8]

materials include aluminum, brass, bronze, copper nickel, and steel. [Ref. 5:pp. 27-29]

2. RAMP Processes

The manufacturing processes required to produce these parts are metal sawing, turning, milling, drilling, tapping, broaching, boring, deburring, and washing. The necessary equipment includes a broaching machine, a horizontal band saw, drill grinders, deburring equipment, a numeric control coordinate measuring machine, and a vertical band saw.

A process plan indicates what steps or processes must be taken, and in what order, to produce a given part. For example, a process plan may be:

- material preparation,
- fixture,
- turning center,
- unfixture,
- fixture,
- horizontal machining center,
- unfixture,
- deburr,
- wash,
- inspect, and
- package.

Fixturing refers to holding or securing a piece while it is being processed. [Ref. 7:pp. 14-25]

Material handling within the RAMP SMP will be accomplished by conveyor, automated guided vehicle (AGV), and manual equipment such as forklifts and hand trucks [Ref. 7:p. 22].

3. Procurement under RAMP

By applying technologies such as CAD and CAM to segments of the procurement cycle and linking the ICP electronically with RAMP manufacturers (including CNSY), procurement administrative lead time and manufacturing administrative lead time can be significantly reduced. When a part is requisitioned, the ICP determines whether the part is available and, if not, whether the part is designated a RAMP part. The ICP will then use an electronic bulletin board to issue the IFB and to make information, such as part specifications, available to RAMP manufacturers. The manufacturers in turn also use the electronic bulletin board to submit bid proposals. A proposal is selected, the contract awarded and production begins. The RAMP manufacturer notifies the ICP when the part or parts have been shipped. The procurement cycle under RAMP is virtually paperless. [Ref. 3:p. 4]

The RAMP SMP Manufacturing System at Charleston Naval Shipyard will be linked with currently existing shipyard functions such as Supply, Central Tool Supply/Tool Management, Equipment and Facility Maintenance, Payroll, and Quality Services. In addition, the RAMP SMP will interface with Navy ordering activities, tooling vendors and cognizant technical authorities. [Ref. 5:p. 7] The Navy ordering activity will transmit six-month forecasts of required repair parts and actual orders for parts to the RAMP SMP. Forecasts of required parts allows for the pre-provisioning of raw materials. It is anticipated that the RAMP SMP system will initially produce 15,000 parts per year with an average lot size of four parts. Shipment of replacement parts will occur on average within 27 days of receipt of the order. [Ref. 7:p. 18]

The required materials and the electronic part technical data (EPTD) must be available at CNSY in order to meet the shipment date. The EPTD describes the part and its attributes and tolerances, and is provided to the RAMP SMP by the Navy EPTD Generation Facility. [Ref. 5:p. 42]

C. FUNCTIONAL COMPONENT DESCRIPTIONS

The RAMP SMP computer system consists of five functional components: Production and Inventory Control, Manufacturing Engineering, Manufacturing, Quality, and Information Management and Communications.

These functional components, their high level interfaces, and the interfaces external to RAMP are shown in Figure 2. [Ref. 8:p. 4] Each functional component represents a cell node of the RAMP network architecture.

Functional components store data in, and retrieve data from, data stores which are databases or files located at the various nodes. Data which are required by only one functional component reside at only one node. Data which must be shared by more than one functional component reside in a common database. Table 2 provides a list of RAMP SMP data stores.

1. Production and Inventory Control

The Production and Inventory Control component performs the following functions:

- Capacity Requirements Planning,
- Production Control,

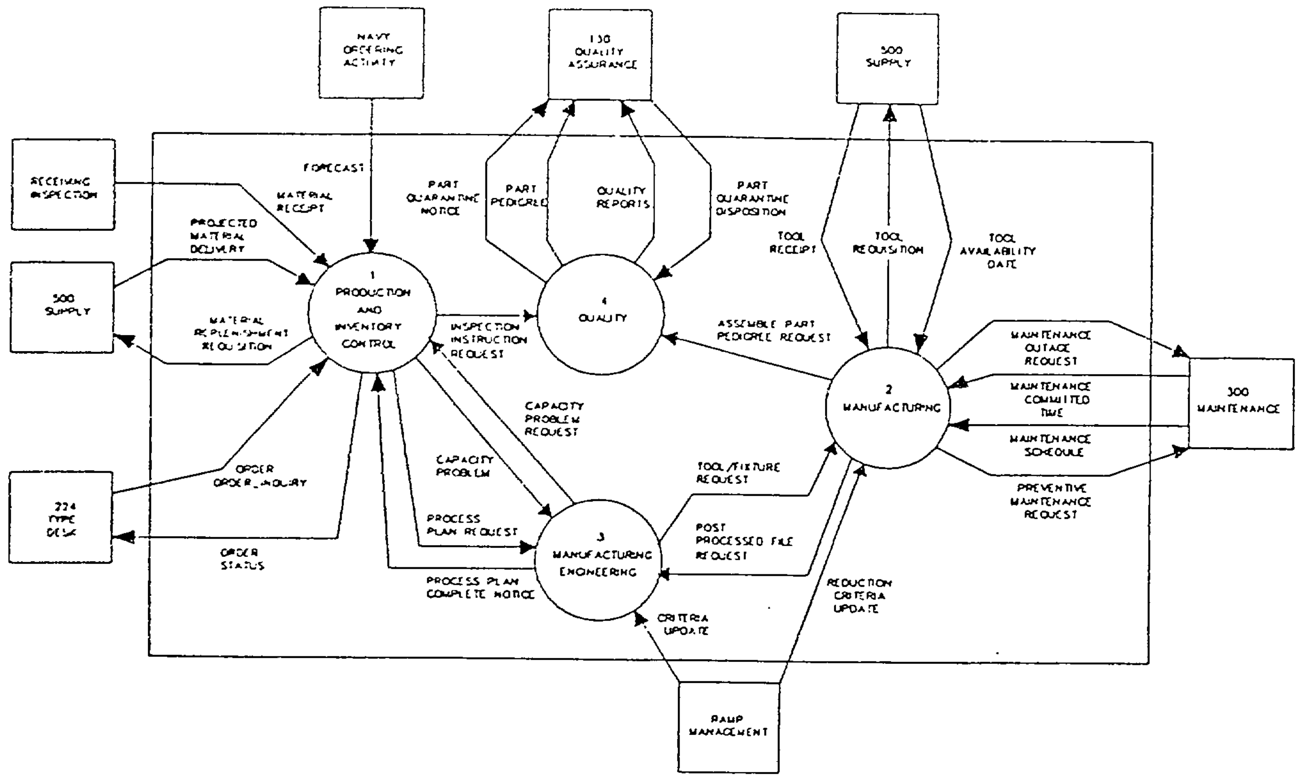


Figure 2. RAMP Internal and External Interfaces [Ref. 5:p. 4]

TABLE 2

RAMP SMP DATA STORES

Common Data Stores

Order	Requisitions
Process Plan	Cell Control
Resources	Quality
Shop Work Order	EPTD
Inventory	Planned Preventive Maintenance
Production Routing	Order Status
Manufacturing Instructions	Post Processed Program
Quality Data	Part Pedigree
Tool Library	

Local Data Stores

Production and Inventory Control

Bill of Material	Order Administrative Data
Group and Sequence Rules	Order Inquiry
Inventory on Order	Forecasted Orders
Long Lead Time Criteria	Pre-provisioned Criteria
Long Lead Time Items	

Manufacturing

Tool Inventory	Active Tool Assemblies
Equipment Operations	Transportation Request
Pallet Assignments	Tool Kits
Reduction Criteria	Work Station Operations
Tool Request Queue	Fixture Inventory
Machine Set Up	Used Tool Listing
Status, State_Data	Non-Perishable Tool Inventory
Reduced Data	Transportation Command Library
Pallet Routing	Cost Data
Maintenance Log	Equipment Specifications
Preventive Maintenance Requirements	

Manufacturing Engineering

Operation Time Standard	Group Technology Code Criteria
Fixture Library	N/C Source
N/C Library	Tool Catalog
Tooling Catalog	

Quality

Pedigree Data

- Order Entry, and
- Material Inventory Management.

Orders for parts are received from the Navy ordering activity and order status data is returned. Requests for the generation of a process plan are sent from Production and Inventory Control to the Manufacturing Engineering component. Requests to generate inspection instructions are forwarded to the Quality component. Production and Inventory Control will convert the EPTD received in the parts order into a format usable by the CAD component. This new format is the Internal Part Technical Data (IPTD) and is written to the IPTD data store. [Ref. 8:pp. 25-63] Production and Inventory Control accesses the data stores as shown in Table 3.

2. Manufacturing Engineering

The Manufacturing Engineering component performs the following functions:

- Check for Repeat Part,
- Code and Classify Part,
- Select Similar Process Plan,
- Revise Process Plan,
- Determine Stock Requirements, and
- Generate Detailed Manufacturing Instructions.

This component determines whether a process plan exists for the part. If one does not exist, a process plan for a part within the same family is selected to serve as the base plan

TABLE 3

PRODUCTION AND INVENTORY CONTROL
DATA ACCESSES

Data Store	Type of Access
Inventory	Read/Write
Resources	Read/Write
Order	Read/Write
Shop Work Order	Read/Write
Bill of Material	Read/Write
Process Plan	Read
Planned Preventive Maintenance	Read
Order Administrative Data	Read/Write
Production Routing	Read
Group and Sequence Rules	Read
Released Shop Work Order	Write
Order Status	Read/Write
IPTD (Internal Part Technical Data)	Write
Order Inquiry	Read/Write
Inventory on Order	Read/Write
Forecasted Orders	Read/Write
Long Lead Time Criteria	Read
Pre-Provisioned Criteria	Read
Long Lead Time Items	Read/Write

for the generation of a new process plan. Manufacturing Engineering notifies Production and Inventory Control when the process plan is complete. From the process plan, detailed manufacturing instructions are generated to determine raw material requirements and to select fixturing configurations and tooling requirements. [Ref. 8:pp. 139-177] Manufacturing Engineering accesses the data stores as shown in Table 4.

3. Manufacturing

The Manufacturing component performs the following functions:

TABLE 4

MANUFACTURING ENGINEERING
DATA ACCESSES

Data Stores	Type of Access
Tool Library	Read/Write
Inventory	Read
Resources	Read
Process Plan	Read/Write
Production Routing	Write
Order Status	Read/Write
IPTD	Read/Write
Manufacturing Instructions	Read/Write
Post Processed Program	Write
Bill of Material	Read/Write
GT_Code_Criteria	Read
Operation_Time_Standard	Read
Fixture Library	Read
N/C Source	Read/Write
N/C Library	Read
Tool Catalog	Read
Tooling Catalog	Read

- Schedules Shop Resources,
- Controls and Monitors the Shop Floor,
- Provides Transportation Control,
- Controls Tooling and Production Equipment, and
- Determines Maintenance Requirements.

Upon receipt of the detailed manufacturing instructions from Manufacturing Engineering, the Manufacturing component generates instructions for the production equipment, including numerical control (NC) source code for NC operations. [Ref. 8:pp. 65-137] Manufacturing accesses the data stores as shown in Table 5.

TABLE 5

MANUFACTURING
DATA ACCESSES

Data Store	Type of Access
Tool Inventory	Read/Write
Tool Library	Read
Inventory	Write
Resources	Read/Write
Process Plan	Read
Planned Preventive Maintenance	Write
Released Shop Order	Read
Order Status	Write
Manufacturing Instructions	Read
Active Tool Assemblies	Read/Write
Equipment Operations	Read/Write
Transportation Request	Read/Write
Pallet Assignments	Read/Write
Tool Kits	Read/Write
Reduction Criteria	Read
Work Station Operations	Read/Write
Tool Request Queue	Read/Write
Fixture Inventory	Read/Write
Post Processed Program	Read
Machine Set Up	Read/Write
Used Tool Listing	Read
Non-Perishable Tool Inventory	Read/Write
Transportation Command Library	Read
Status, State Data	Read/Write
Reduced Data	Read/Write
Pallet Routing	Write
Quality_Data	Write
Part Pedigree	Write
Cost Data	Write
Equipment Specifications	Read
Maintenance Log	Read/Write
Preventive Maintenance Requirements	Read

4. Quality

The Quality component performs the following functions:

- Generate Final Inspection Instructions,
- Determine Disposition of Quarantined Parts,

- Analyze and Report Quality Data,
- Assemble Part Pedigree, and
- Validate Part Manufacture.

Upon receipt of a request from Production and Inventory Control, the Quality component generates the final inspection instructions. When a shop work order is completed by Manufacturing, a part pedigree is assembled. The pedigree data contains all information concerning the raw materials and the equipment operations performed on the raw materials. The pedigree data, part specifications and inspection results are evaluated. [Ref. 8:pp. 178-187] The Quality component accesses the data stores as shown in Table 6.

TABLE 6

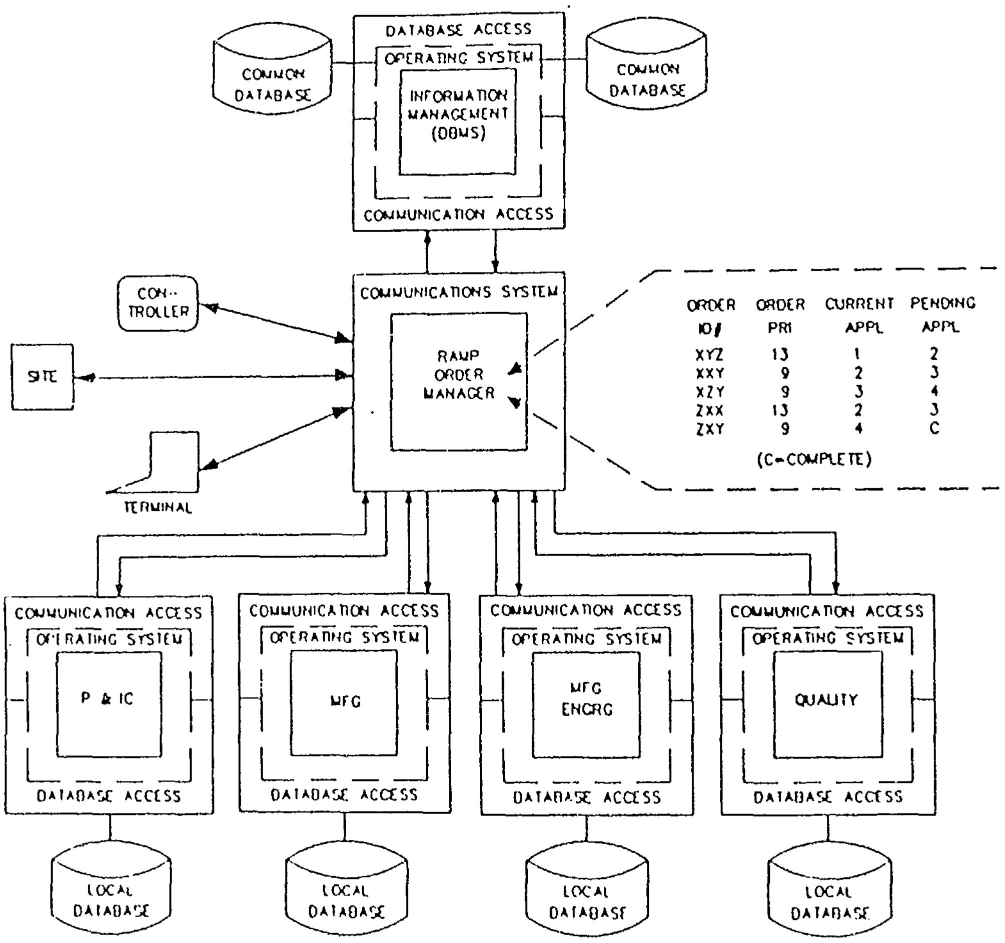
QUALITY DATA ACCESSES

Data Store	Type of Access
Order Status	Write
IPTD	Read
Manufacturing Instructions	Write
Quality Data	Read/Write
Part Pedigree	Read/Write

5. Information Management and Communications

Information Management and Communications provides the network to link all of the other functional components. The functional components store data in local databases and up load the data to a common database when requested. When data are needed by a functional component to perform its part in the production process and are not available locally, the data are requested through Information Management. This functional component provides the database management system (DBMS) to integrate the various databases and provide the shared data needed by the application programs. Local databases and the common databases are connected as shown in Figure 3. [Ref. 9:p. 2.0-L3] Local databases may or may not be relational depending on the environment and requirements at that node. The functional components (Production and Inventory Control, Manufacturing Engineering, Manufacturing, and Quality) are comprised of heterogeneous off-the-shelf application programs written in high order languages such as COBOL, FORTRAN, and C languages. [Ref. 8:p. 188] The DBMS of the Information Management component provides the access to the databases necessary to an application through logical user views. Figure 4 shows the Information Management component at the center controlling (by the DBMS) access to the shared data stores.

Also in Figure 4 is the Communications component shown around Information Management. It provides the links to the functional components (and their applications) and the links to functions external to RAMP (Navy Ordering Activity, Type Desk, RAMP Management, Supply, Receiving Inspection, Quality Assurance, and Maintenance). The



ORDER ID#	ORDER PRI	CURRENT APPL	PENDING APPL
XYZ	13	1	2
XXY	9	2	3
XZY	9	3	4
ZXX	13	2	3
ZXY	9	4	C

(C=COMPLETE)

Figure 3. RAMP Network Connecting Common and Local Databases [Ref. 9:p. 3.2-R20]

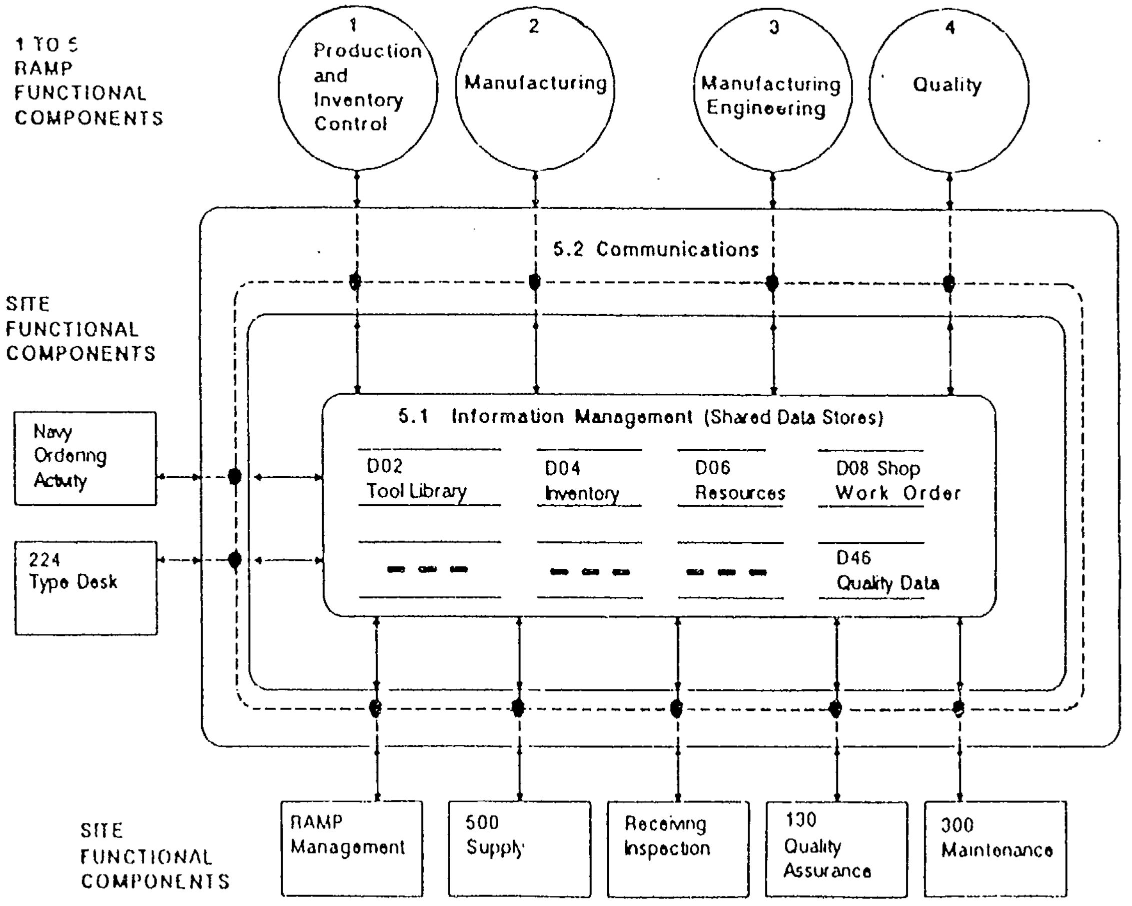


Figure 4. RAMP Top Level Communications System Model [Ref. 8:Appx. III, p. 17]

Communication System will provide file and message exchange services for the application functions, data handling, links to other networks, and data communications management. [Ref. 8:pp. 193-194]

While local data will be managed at the local level, the common database is managed by the DBMS provided by Information Management as shown in Figure 3. This DBMS will have the following capabilities:

- Applications Interface to RAMP System Database,
- On-line/Interactive Data Dictionary,
- On-line/Interactive Applications Development,
- On-line/Interactive Report Generator,
- Database Recovery, and
- Database Security. [Ref. 8:p. 188]

The data dictionary will contain data element and relational data model definitions as well as ownership and security data. Structured Query Language (SQL) commands will be processed by the Data Dictionary Function which will create and maintain the schemas, the logical user views, data control declarations to restrict access, and security declarations. [Ref. 8:p. 189] Chapter IV will examine the use of an Information Resource Dictionary System to facilitate the implementation of the dictionary and directory functions in the RAMP architecture.

The Applications Development Function provides the user with the capability to use and modify terminal displays with an interactive "screen painter." The user may also create applications with access to databases. Using SQL statements, the user may generate reports, forms and mailing labels through the Report Generator Function. [Ref. 8:pp. 189-190]

The Database Recovery Function protects against user program failure, system hardware or software failure and disk media failure. Before-and-After images of updates to the common database will be recorded on the recovery log. "Roll Back" and "Roll Forward" will be used to recover the common database. [Ref. 8:p. 191]

Use of the RAMP common database will be controlled by the Database Security Function which will process SQL statements granting or revoking access privileges to tables and logical user views. [Ref. 8:pp. 191-192]

D. CONTROL SYSTEM

Each device or piece of equipment and each process necessary to produce a part in the RAMP SMP Manufacturing System is controlled by its own individual software or application package. It is anticipated that this individual software will be commercial off-the-shelf (COTS) software. The discussion which follows describes the control architecture used to coordinate RAMP resources in this environment of distributed COTS applications.

1. Control Hierarchy

Control within the RAMP SMP Manufacturing System is hierarchical (Figure 5). The RAMP functional components (Production and Inventory Control, Manufacturing Engineering, Manufacturing, Quality, Information Management and Communications) are at the cell level in the hierarchy. [Ref. 7:p. 33]

Workstations are controlled by the cell controllers. One or more devices or pieces of equipment on the factory floor comprise a workstation. Each workstation controls one or more device controllers. Local databases and DBMSs exist at the workstation level as well as at the cell level. [Ref. 7:p. 34]

At the lowest level are device controllers which control one or more devices or equipment such as numerically controlled machine tools. Equipment level controllers do not have associated databases but rather depend on the workstation controller for data. [Ref. 9:p. 3.1-9L]

The RAMP SMP functional components reside at nodes of the network. A network view of the RAMP cell, workstation and device control system is shown in Figure 6. Specific equipment is indicated by D1, D2, D3 and is controlled by the device controllers.

2. RAMP Order Manager

Data which is required only by a single functional component or by a workstation will reside in a local

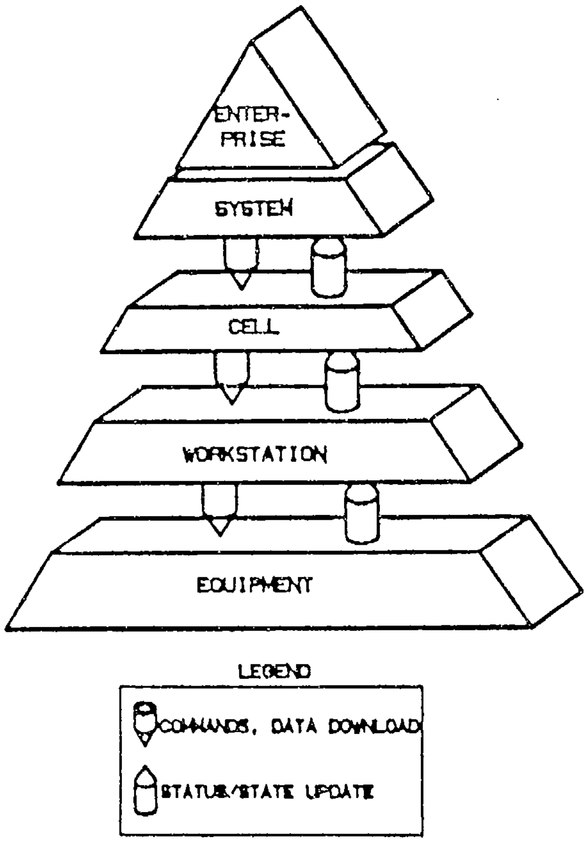


Figure 5. RAMP Hierarchy [Ref. 9:p. 3.1-7L]

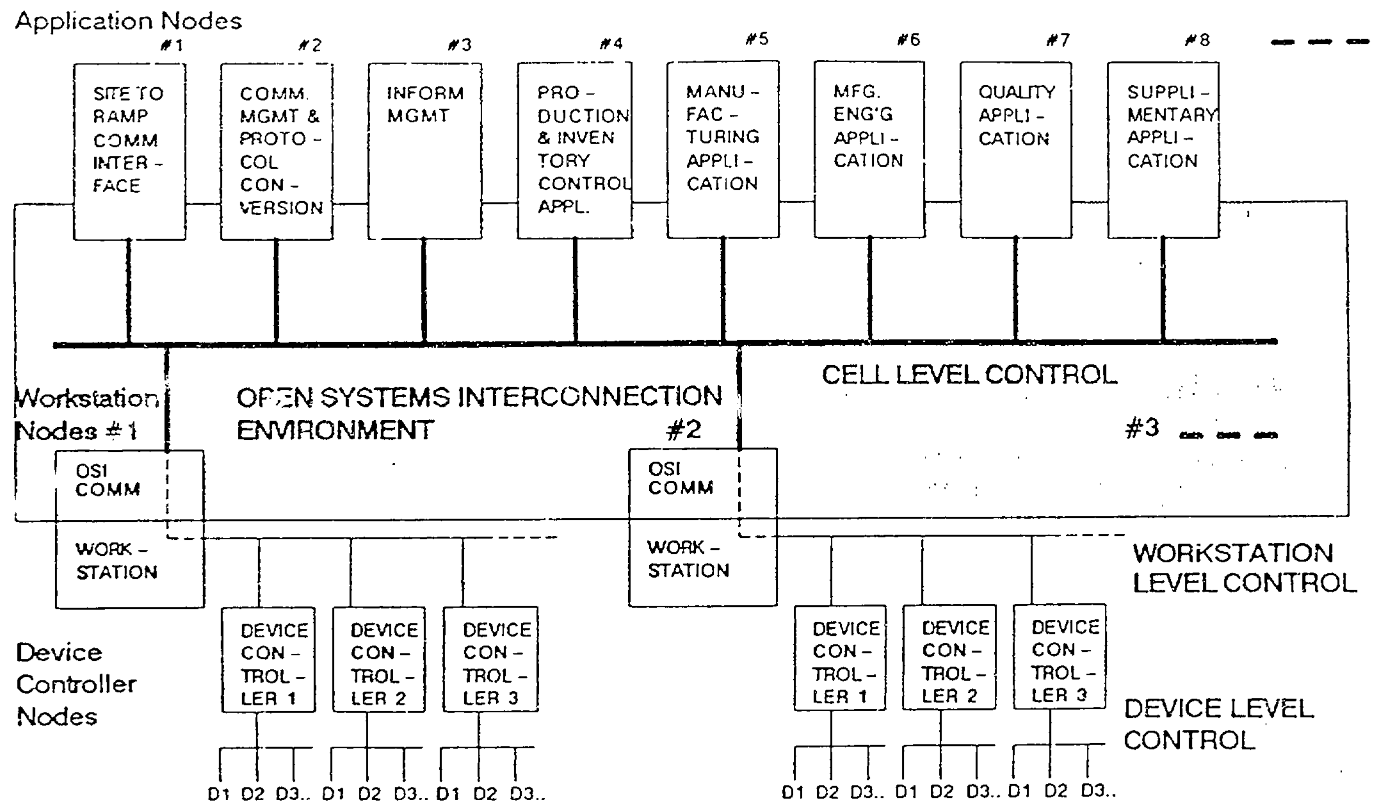

Application Nodes
#1
SITE TO RAMP COMM INTER - FACE
#2
COMM. MGMT & PROTO - COL CON - VERSION
#3
INFORM MGMT
#4
PRO - DUCTION & INVEN TORY CONTROL APPL.
#5
MANU - FAC - TURING APPLI - CATION
#6
MFG. ENG'G APPLI - CATION
#7
QUALITY APPLI - CATION
#8
SUPPLI - MENTARY APPLI - CATION
CELL LEVEL CONTROL
OPEN SYSTEMS INTERCONNECTION ENVIRONMENT
Workstation Nodes #1
#2
#3
OSI COMM
WORK - STATION
WORKSTATION LEVEL CONTROL
Device Controller Nodes
DEVICE CON - TROL - LER 1
DEVICE CON - TROL - LER 2
DEVICE CON - TROL - LER 3
DEVICE LEVEL CONTROL
D1 D2 D3..

Figure 6. A RAMP SMP Network Model [Ref. 8:Appx. III, p. 18]

database. The local databases are tied to the COTS packages and may be relational in structure or simply file structures depending upon what the COTS supports. All data which needs to be shared by more than one functional component will reside in a common database. This common database will be managed by the DBMS provided by the Information Management component. RAMP must integrate and control these devices and processes and the sequence in which the processes occur. Data from the common database necessary to an application must be transferred to and from the local database.

The RAMP Order Manager (ROM) provides the centralized management of all RAMP applications software and the processing of all requests. The ROM exists at the System level of the RAMP Control Hierarchy shown in Figure 5. The ROM software will be developed in the C language. This software will be completely independent of the commercial off-the-shelf software used by the manufacturing equipment or the applications [Ref. 9:p. 3.3.1.2].

The ROM is event driven which means it issues commands to, and receives status reports from, the functional components. There are three functional components of ROM: ROM Manager, Process Manager and Application Manager. Figure 7 shows the relationships between them. The ROM Manager coordinates ROM processing, accepts all status data and routes the status data to the

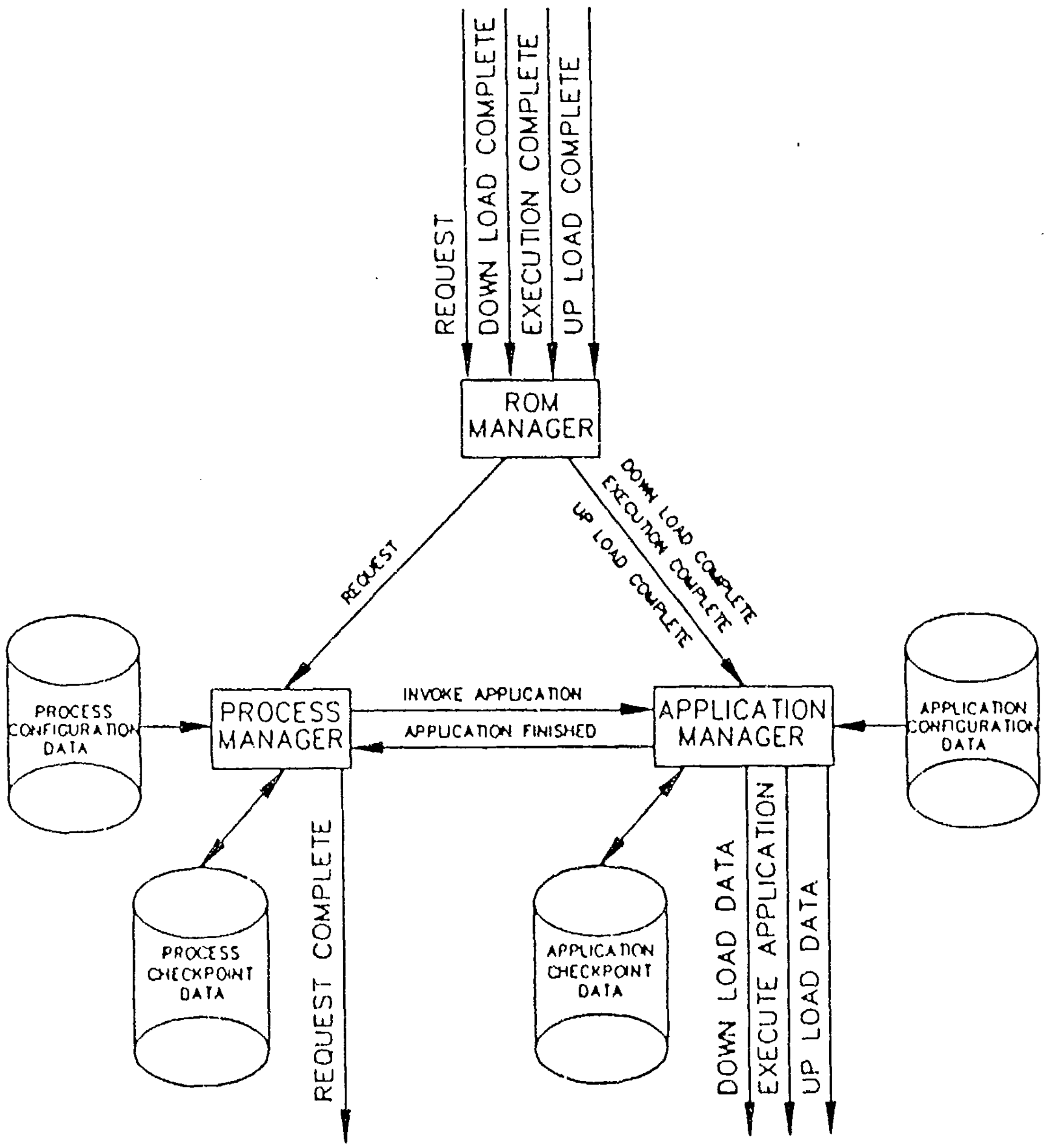


Figure 7. ROM Functional Components [Ref. 9:p. 3.3.1.2-L8]

appropriate ROM component, either the Process Manager or the Application Manager. [Ref. 9:p. 3.3.1.2-R9]

The Process Manager processes requests received from the ROM Manager, such as orders or inquiries, and manages the sequence of application packages necessary to complete a request. Each sequence is considered a "process." For example, a process may include the following applications:

- initiate order,
- create process plan,
- determine projected item delivery,
- reserve capacity,
- process plan error recovery, and
- reject order.

When a request status is received, the Process Manager determines the process necessary to satisfy the request and the first application in that process. In this example, the first application is "initiate order." It then requests the Application Manager to invoke the first application. When the Application Manager responds that the application is completed, the Process Manager determines the next application, in this case "create process plan," and requests the Application Manager to execute it. If the Application Manager had responded with a status indicating that the "initiate order" application did not complete, the Process Manager would have determined the next application to be "reject order." When the entire sequence or process is completed, the Process Manager issues a Request Complete status. [Ref. 9:p. 3.3.1.2-R10-R11]

When a request to initiate an application is received from the Process Manager, the Application Manager checks for the availability of that application and, if available, issues a command to down load the necessary data to the appropriate RAMP functional component. A "Down Load Complete" status is received from the ROM Manager at which time the Application Manager issues the command to execute the application. When the application has executed successfully, the Application Manager issues a command to up load the data to the common database. The ROM Manager informs the Application Manager when the up load has completed. The common database is now updated and current. At this point the Application Manager informs the Process Manager that the application execution has completed. [Ref. 9:p. 3.3.1.2 R13-L16]

The ROM maintains configuration tables or decision tables to define both processes and applications. In the previous example, the Application Manager returned a status message to indicate whether or not the "initiate order" application successfully completed. The Process Manager consults a decision table to determine the next application, in this case either "create process plan" or "reject order." These decision tables are maintained as status tables which are initialized from the configuration tables. As each status is received, ROM checks the status table to determine the command to be issued. Checkpoint files are updated after every status message to provide redundancy to the status tables. This redundancy also allows the system to be restarted from the checkpoint files. [Ref. 9:p. 3.3.1.2-R3]

E. EXTERNAL INFORMATION INTERFACES[1]

The RAMP SMP Manufacturing System installed at CNSY will be linked with the following shipyard functions: Type Desk, Supply, Receiving Inspection, Quality Assurance, Maintenance, RAMP Management, and Navy Order Activity.

The Type Desk will send Order and Order Inquiry messages to Production and Inventory Control. Order Status messages will be returned. An Order message will contain the Electronic Part Technical Data.

Material Replenishment Requisition messages from Production and Inventory Control to the Supply department will contain the Job Material List. The Supply department will return a Projected Material Delivery message.

When material is received at the shipyard and is available to RAMP, a Material Receipt message will be sent from the Receiving Inspection to Production and Inventory Control.

The RAMP Quality component will send a Part Quarantine Notice message to shipyard Quality Assurance when a part

[1]Information for this section was drawn from [Ref. 10]

fails a quality inspection. A Part Pedigree message will be sent when a part has passed inspection and is ready to be shipped. Quality Report messages from RAMP Quality to shipyard Quality Assurance will summarize production quality. Shipyard Quality Assurance will send Part Quarantine Disposition messages to the RAMP Quality function in order to provide instructions for the disposition of parts which fail to pass inspection.

The RAMP Manufacturing component sends a Tool Requisition message to the Supply department when tools or fixtures are needed. The Supply department will respond with a Tool Availability Date message. When the tool is shipped to RAMP, a Tool Receipt message is sent from Supply to the Manufacturing component.

A Maintenance Outage Request message is sent from RAMP Manufacturing to shipyard Maintenance whenever immediate maintenance is needed due to equipment failure. Maintenance Committed Time messages are returned to the Manufacturing component and indicate the date and time for which repair is scheduled to begin and end. Preventive Maintenance Request messages from Manufacturing initiate the Maintenance Schedule messages from shipyard Maintenance.

RAMP Management, a shipyard function, informs Manufacturing of changes in data collection requirements by a Reduction Criteria Update message. Criteria Update messages from RAMP Management to Manufacturing Engineering allow for changes, such as part classification, to be made to that functional component.

In order to provide for sufficient inventory of material, Forecast messages containing raw material requirements will be sent from the Navy Ordering Activity to the Production and Inventory Control component.

F. SUMMARY

Procurement of spare parts has long been a problem for the Navy. Long lead times in the procurement cycle impair fleet readiness. The RAMP SMP Manufacturing System is a flexible manufacturing system designed to produce a wide variety of low volume parts on demand. On-hand inventories are reduced and at the same time fleet readiness is improved.

Both automated and manual processes will be present in the RAMP SMP. CAD/CAM and Computer-Aided Process Planning (CAPP) systems are integral to the RAMP SMP. The RAMP control system is hierarchal. Operations are computer controlled and event driven. RAMP internal processes store data in and retrieve data from both local databases and a common database. Because the RAMP SMP Manufacturing System will be installed at the Charleston Navy Shipyard (CNSY), interfaces with CNSY departments must be established.

There is a wide variety of equipment, much of it computer controlled, physically distributed throughout the factory floor. The physical and functional systems of RAMP must be integrated through a control system to provide a complete manufacturing unit. The next chapter will take a close look at some of the complexities inherent in a distributed environment such as RAMP. In particular, the data administration issues which exist in Computer Integrated Manufacturing and Flexible Manufacturing Systems will be examined. The ramifications of these issues in the RAMP system will then be discussed in Chapter IV.

III. DATA ADMINISTRATION IN DISTRIBUTED COMPUTING ENVIRONMENTS

A. INTRODUCTION

This chapter provides the conceptual foundation for understanding the data administration aspects of the Navy's RAMP project to be presented in Chapter IV. Section B establishes the need for distributed processing in a flexible manufacturing environment. Section C explores the data administration issues that must be addressed in such a system. Section D presents a data administration architecture developed by the National Bureau of Standards to manage FMS environments. Section E concludes the chapter with a summary of the salient distributed data issues in FMS.

B. DISTRIBUTED PROCESSING AND FLEXIBLE MANUFACTURING

In order to gain an appreciation for the complexities involved in managing a flexible manufacturing or computer integrated manufacturing environment, this chapter begins by defining distributed processing. It then establishes the need for distributed computing in this type of manufacturing technology by briefly discussing the major functions involved in the process.

1. Distributed Computer Systems

Distributed computing is a relatively new field, which is experiencing explosive growth:

> In recent years there has been a dramatic fall in the cost of hardware processors and memory, combined with similar cost reductions and technological advances in the communications field. The result has been to make computer networks and interconnected computer systems a viable and cost effective solution in many environments. [Ref. 11:p. 1]

While there is no accepted definition of distributed computer systems, they generally involve some degree of distribution of processing, control, and/or data. Distributed systems imply multiple processing units but the mere physical separation of these components is not indicative of a distributed system. There must be functional partitioning of processors as well as interaction among the various functional components, preferrably while providing a single system image. These characteristics are summarized below:

> A distributed processing system is one in which several autonomous processors and data store supporting processes and/or databases...cooperate to achieve an overall goal. The processes coordinate their activities and exchange information by means of information transferred over a communications network. [Ref. 11:p. 4]

The three general types of distribution are briefly discussed in the following sections.

a. Distributed Processing

Distributed Processing refers to the manner in which the hardware of the system "processes" the various tasks to be performed. There must be at least two computers, each with its own local memory and processor. This reflects not only physical distribution, but may also provide function distribution, where each of the different computers performs its own function of the complete process. An essential element here is some form of communications network which enables these distributed computers to cooperate with one another. Without this cooperation to achieve a common goal there would be no distribution, but rather two isoloated computers performing their own independent tasks.

b. Distribution of Control

There are different ways of providing coordination among the various components of the system. One method is to use a centralized strategy, where one control mechanism "controls" interaction among both the physical and logical resources of the system. If control is distributed, however, one approach is to use a sort of hierarchy, with each level responsible for a particular area of coordination. Still another method of providing distributed control is to allow all of the individual processors complete autonomy over their local resources. In this case an additional goal of the system is to provide some degree of "transparency" in order to mask the physical distribution and heterogeneity of its components.

c. Data Distribution

Perhaps the single most important resource requiring control in any computer system is the data. Data can be distributed across the system in various ways. Replicated data provides individual copies of all of the data at every node in the system. Data that is partitioned stores different pieces of data at various components of the system. Hybrid data distribution is a data placement scheme that does a little of both. In RAMP, data that are needed by only a single functional component are stored in a local database while data that needs to be shared among the various components are stored in a global database.

One of the major advantages of distributed systems is the increased degree of flexibility provided. In order to provide interaction among the various components, distributed systems must be constructed in a very modular manner, with well-defined interfaces. Because of this requirement, properly designed distributed systems have the added advantage of being able to adapt to technological or software changes without significantly affecting the other components of the system. This flexibility is a major requirement in the computer integrated manufacturing (CIM) environment discussed below.

2. <u>Distribution and Computer Integrated Manufacturing</u>

Distributed computer systems require significant partitioning and interaction among the functional components and tasks of the system. This is complex enough when the system is composed of homogeneous hardware and/or software, but even more problematic in flexible manufacturing systems (FMS) where the environment encountered is more likely one containing heterogeneous components. Beeby identifies the substantial advantage of heterogeneity, however: "new computing products can be incorporated as they become available." [Ref. 12:p. 103] This philosophy is exemplified by RAMP's use of commercial off the shelf (COTS) application packages which control the local databases. As newer, more efficient software packages become available, they can be substituted with minimal disruption to operations.

Davis et al., identify the following functions which must be integrated in a CIM environment:

- marketing and sales,
- manufacturing data preparation,
- production planning and inventory control,
- production scheduling,
- process supervision, and
- quality assurance. [Ref. 13:p. 1]

The system must additionally provide the tools necessary for:

- defining the necessary information,
- inputting the necessary information,
- locating and accessing the necessary information,

- communicating the necessary information,
- displaying the necessary information, and
- updating the necessary information. [Ref. 14:p. 78]

A brief synopsis of CIM functions and their RAMP counterparts is given below.

a. Marketing and Sales

The primary interface between a customer and a manufacturing facility is the marketing and sales function. Data required here centers around available products, including price, specifications, and delivery schedules. Before new products are introduced, this function determines their profit potential. The data required here can be generally classified as administrative. This function currently resides external to RAMP.

b. Manufacturing Data Preparation

This function is equivalent to the "Manufacturing Engineering Function" in RAMP. All of the functions necessary to generate the data required in the manufacturing of a part or product are included in this function. For example, the engineering department translates the customer's requirements into product designs which include "detailed three dimensional drawings, geometry data, tolerances, and other required manufacturing specifications." [Ref 13:p. 3] Computer Aided Design (CAD) and Computer Aided Engineering (CAE) tools are often involved. This function is further complicated by the fact that different engineering functions have different data requirements. The data required for manufacturing can be categorized as highly technical and math-intensive, in sharp contrast to the administrative data required by the previous function.

c. Production Planning and Inventory Control

Called "Production and Inventory Control" in RAMP, this function is responsible for developing a list of the actual jobs to be completed over a specified period of time (next month, for example). All of the hardware such as n/c machines required for drilling, boring, metal sawing, turning, broaching, and deburring of the scheduled products are listed. The inventory control function ensures that all raw materials needed for the process will be available. Again, the data needed to generate these detailed production plans bear little resemblance to the data required in previous functions.

d. Production Scheduling

This is the function which controls the daily workload of the enterprise. Detailed schedules are developed for the operations required to complete the jobs issued by the detailed production plan created in the previous section. The exact sequence of tasks to be performed in the production process, as well as anticipated start and finishing times are assigned. Extensive interprocess coordination provides continuous monitoring of the feedback from process supervisors to ensure that all raw materials are in place, and machine tools are available. [Ref. 13:p. 4] The data required here must be accurately generated from previous functions of the process. Production Scheduling in conjunction with Process Supervision comprise the "Manufacturing" component found in RAMP.

e. Process Supervision

The process supervisor serves two main functions. First, he implements the instructions from the process plan for every operation. He then monitors the process to ensure the actual compliance with the instructions. He may intercede to slow down or speed up a machine tool, or make other minor changes in the processing. Any major deviations, however, will require additional intructions from previous functions.

f. Quality Assurance

Termed "Quality" in RAMP, the final function in this process is quality assurance which is divided into two main areas. First, output from each process is verified to ensure that it meets the specifications provided by the manufacturing data preparation function. When errors are found, this information is used to correct problems in the designs, process plans, and the actual processes. The second function is used to predict machine maintenance and replacement requirements. This information relates to actual machine performance. Historical data related to the product such as CAD designs, process plans, inspection and machining procedures, and materials used are also kept by this function.

C. DATA ADMINISTRATION CONSIDERATIONS

This section motivates the need for data management, and then details specific data administration issues which must be considered for CIM.

1. Data Management

The CIM environment is a heterogeneous one containing multiple databases and data types, contained in different hardware configurations, and accessed by a wide variety of users and applications. These complexities are compounded by the requirement for simultaneous access and update capabilities. The more complex the environment, the more crucial the data management function. Inaccurate databases or repeated down-time will rapidly undo the benefits of FMS so data management is a function that must be given special consideration.

Data management is further complicated by the individual characteristics of the data required by the functions discussed in the previous section. Part models and process plans, for example, may be accessed and updated several times a month and require several kilobytes of storage each. Equipment status data on the other hand, requires continuous updating on only a few hundred bytes of data. These factors impact data management strategies in various ways.

> It is a major factor in the decision to distribute or centralize data. It influences the choice of topology, protocol, and packetizing strategy for the network...and plays a prominent role in scheduling responses to data requests and for performing query optimization. [Ref. 13:p. 10]

The RAMP environment is highly heterogeneous with each application having its own local data store and thus different capabilities for sharing and accessing data. There must therefore be an underlying mechanism to disguise the differences among these various data environments. This is accomplished through generation of local conceptual schemas which define logical relationships among data elements contained in local databases, a universal data definition language which defines those schemas, and finally a data manipulation language to be used for data storage and retrieval. A global data model defines how the local data are related and a directory indicates where data are distributed throughout the system.

There must be a mechanism for a user to access the global database. A user should not be required to know exactly how or where a particular piece of data is stored; he should merely be able to request it and have it delivered in the format required. This requires a global data manipulation language (DML) as well as local DMLs for the local databases.

Finally, there must be a mechanism for maintaining the integrity of the databases. These issues are the topic of the next section.

2. Data Administration

Previous sections discussed the diverse nature of the data found in computer integrated manufacturing. What needs to be done to guarantee, however, that the data required by each function is in fact the correct data? How can CIM ensure that only authorized users, be they people or other computers, have access to the data needed. What needs to be done to accommodate the simultaneous requests by users for access to the same data resource? What mechanism exists to restore the contents of the databases in the event of hardware or software failure? Finally, what type of mechanism exists to "facilitate the data sharing, reduce data redundancy, and provide an integrated environment for data manipulation"? [Ref. 15:p. 48] In Data Administration (DA) terminology, these questions are addressedby the following policies:

- database integrity,
- database security,
- database update and concurrency control,
- database back-up and recovery,
- information resource management.

a. Database Integrity

One of the major issues of the integrity function is that of ensuring that the values that are in the database are only those values that are allowed to be there. This function of validity involves ensuring that only acceptable data values are entered into the database. However this only guarantees that the values fall within a predefined range, not that the database contains accurate values. Other aspects of data integrity attempt to ensure that data accuracy is maintained.

b. Database Security

Database security addresses the issue of allowing only authorized users access to data. This includes access resulting from both innocent mistakes such as keying errors and deliberate attempts to penetrate the system.

Access and Capabilities lists assist in the security function. These lists control who is allowed access, and what they are allowed to do once access is granted. They identify the resources that require protection, and all individuals/terminals/programs that require access to those resources. Passwords are another common form of security mechanism.

Access can be controlled by individual, specific hardware device, application, time of day, department, etc., or any combination thereof. For example, production engineers may be granted access to all database fields relating to actual production of a part, but restricted from accessing any financial information required by the marketing department. They may have full capabilities including update, add, delete, calculate and reading rights, or they may be restricted to one or more of the capabilities only. Establishing the appropriate view is a vital step in the database design process.

In a distributed environment there is the additional security concern of ensuring that data will not be intercepted by an unauthorized user while it is traveling across the network. If the system is penetrated, data encryption renders the data meaningless to the penetrator without the decryption code. Two types of encryption are possible. In the first "data is encrypted, transmitted through the communications system, and decrypted at the destination node." A second type of encryption mechanism is one "which actually encrypts the data before it is stored on the database." The most secure system employs both methods. [Ref. 16:p. 94]

c. Database Update and Concurrency Control

One of the main purposes of database management is to allow multiple users to share data. No problem arises as long as they are accessing the data for 'read-only' purposes. This we know is not the case in the CIM environment, and other measures must be taken to prevent the following from occurring:

- two transactions are simultaneously updating the same data item,
- one transaction is reading an item while another is updating the same item,
- two transactions requiring the same data items are waiting for each other (deadlock),
- one transaction is continually preempted by others requiring the same items (livelock). [Ref. 13:p. 12]

These issues are further complicated in an environment like RAMP which uses either replicated or hybrid data distribution. With multiple copies of the database, consistency and integrity must be maintained within a single database, as well as among the various copies of the database. Concurrency control mechanisms to address these potential problems are generally based on a two-phase locking or timestamp ordering approach.

When the locking approach is used, flags are used to "lock" specific data items. These items can range from an individual data attribute, to a particular record, or even to the entire database. Necessary operations are performed on the data and the flags reset to indicate that the locks have been released. There are various locking approaches. Some, such as the global locking approach, ensure that all copies of the database are in agreement all of the time. This approach requires significant communication overhead, however. Others force a reduced level of consistency where at a given time all copies of the database may not be consistent. Additionally, while these approaches address the simultaneous read and update issues, they do nothing to prevent deadlock and livelock from occurring.

Timestamp ordering techniques assign a read and a write timestamp to each transaction. A problem arises in a distributed environment, however, "since each node has its own internal clock, and it is very difficult to synchronize them precisely to ensure that there is a single network-wide clock." [Ref. 16:p. 126] Although timestamps assist in eliminating deadlock and livelock, there is "considerable overhead involved in storing timestamps for all the data items in the databases." [Ref. 13:p. 13] The problem in RAMP is that we may have local data environments with different concurrecy mechanisms (timestamping in one; two-phase locking in the other, for example). The RAMP Order Manager (ROM) must ensure that these environments do not conflict and compromise data integrity.

d. Transaction Granularity

Transactions or logical units of work (LUW) are predefined segments of processing which, when completed cause the physical database(s) to be modified. Until an LUW is actually completed, all changes made to the database during that time are stored in temporary buffers rather than the database itself. In RAMP a transaction could be defined as an entire process, i.e., production of an entire part, or simply as the delivery of a partially completed part to its next workstation. In either case locks are employed to prevent other processes from updating any data items which the transaction accesses until the transaction has been committed, or logged as completed.

Suppose a transaction were defined to encompass the entire production of a part. If write locks were applied, no other process would be able to access any of the part-related database items until after the entire part processes were completed. Although certain phases would have actually been completed, their related databases would not reflect this status since no database commits take place until the entire transactionhas been completed. Further, other processes would continue to be locked out unnecessarily when access really could have been granted. This may degrade overall system performance unsatisfactorily. If, on the other hand, the transaction granularity were defined too small, a large number of commits would be made to the databases, access to data resources would increase, but database recovery would now become significantly more difficult if the process aborted before completion. Both situations affect database recovery strategies.

e. Database Back-up and Recovery

There are three major functions involved in the back-up/recovery function. First, data must be retained for use in recovering an operation which aborts. This includes saving and storing all of the data needed in the event a failure occurs. Second are the procedures themselves which enable the database to be restored. Third are control procedures to be used to ensure that once the database is restored its integrity is maintained.

Failures can occur for various reasons:

- read/write heads scratching the disk surface,
- power supply is interrupted,
- operating system error,
- operator error,
- program or transaction abnormal ending (abend),
- manufacturing and/or computing hardware failure.

Whatever the cause, it is critical that recovery procedures be initiated. A single production process failure may require that multiple databases on multiple machines be restored. Before-and-after images and checkpoints can be used to restore the databases to their states prior to a failure, but not without considerable overhead. Maintaining before/after images extracts a serious efficiency penalty since each transaction must be entered into the image journal as well as into the databases themselves. Journals take additional disk space, and could actually prevent some of the time critical requirements of the CIM environment from being met.

f. Information Resource Management

A final area of data administration is an integrated information resource management tool for facilitating data integrity and monitoring database efficiency. This tool is called an information resource dictionary system (IRDS).

An IRDS is a "logically centralized repository of data about all relevant information resources within an organization." [Ref. 15:p. 49] It contains the logical data models for each database, as well as the physical information required to access them. The data dictionary component is used to record, store and process information related to the logical aspects of the system. This metadata or "data about data" is not concerned with the validity of the data within the databases; its sole function is to provide a description of them. For example, suppose we have a file in a CIM database called PARTS. The dictionary component would have such information as a listing of the fields within the file (part_no, part_name, etc.), where the data comes from, particular integrity constraints which must be adhered to, which programs access the file, and so on. The actual values for part_no, part_name, etc. would not reside in the dictionary, however, but rather in one or more operational databases. Note that this type of information is logical in nature.

The directory component of the IRDS maintains all metadata about the physical aspects of the system. This includes information relating to the actual physical location of the data and the methods or resources which access them. The directory component which relates to the PARTS file in the example above would contain information on such aspects of the file as where the file is physically stored, what file structure is employed,and what operating systems are used.

An IRDS can be further characterized as being active or passive, depending upon the "scope of control exercised through metadata management." [Ref. 17:p. 21] If a program or process depends upon the metadata provided by the IRDS in order to perform its function, the IRDS is active. If, on the other hand, that same program can obtain its metadata from other sources outside the IRDS, the IRDS is passive. The IRDS may, in this case, simply document the metadata after the fact.

An IRDS is particularly necessary in a distributed environment and may itself be distributed. Regardless of the data distribution scheme employed, an IRDS is required to "support the development and operation of distributed databases." [Ref. 17:p. 56] If data are replicated, the IRDS must keep track of all of the known redundancies throughout the system. When data are partitioned, the IRDS must know how all of the various pieces fit together to form the "whole" database, in a manner which is transparent to the user.

Particularly in the heterogeneous CIM environment, locating the required data is only half the problem. Once found, the data will more than likely undergo a complex process of data translation before it gets to the requestor. Upon return, this same translation process is required once again. Since both hardware and software in a heterogeneous environment may differ from node to node, this translation involves not only individual "languages" (such as a translation from ORACLE to INGRES), but also machine specific translations (such as from ASCII to EBCDIC, or a 32-bit word to a 16-bit word). When the DBMS's themselves are different, as is more than likely the case, a logical translation of the data structures is also necessary. The IRDS facilitates the translation process by storing metadata mappings and access paths to allow the source to be transformed into the target data [Ref. 17:p. 236].

We now review a data administration architecture developed by the National Bureau of Standards (NBS) that addresss the problems we have discussed in the distributed computer integrated manufacturing environment.

D. AN ARCHITECTURE FOR DISTRIBUTED DATA MANAGEMENT IN CIM

Previous sections described how flexible manufacturing and computer integrated manufacturing systems will require a network of heterogeneous hardware/software systems. An additional problem encountered in this type of environment is the requirement for real-time access. This section is devoted to reviewing a prototype developed by (NBS) to facilitate the data sharing requirements of the CIM environment.

1. Background

As discussed above, a fundamental characteristic of the CIM enviroment is the diversity of computer systems:

> Rarely does the same kind of computer perform engineering support, real-time control and administrative applications. Consequently, data sharing is complicated by differing operating systems, hardware architectures, data systems and access methods. To get the same information from two different machines, expect to use two different interfaces, phrase the request in two different 'languages' and get back the information in two different forms. Since data systems range from very limited to very powerful, it is possible for one of two data systems to lack a capability that another has. [Ref. 18:p. 44]

Since data in the CIM environment must be shared extensively among all of the different components, and in light of the requirement for heterogeneity, more often than not multiple databases are created with some degree of overlap. An update to one therefore necessitates an update to others. Manufacturing Automation Protocols (MAP) were created to standardize the interchange of files between any two computer systems [Ref. 19:p. 66]. This is not sufficient for the CIM environment, however, since simple files must be first extracted from complex databases before the protocol can be employed. To accomplish this, application programs would have to "know" not only the location of the data they use, but also their organization. Since these two factors are constantly changing as new applications and equipment are added, this is not a practical option.

A better approach is to provide a common interface between programs and databases. A change in architecture would require only a change to the interface, not to the applications. This is the premise upon which the NBS prototype is built.

2. The Integrated Manufacturing Data Administration System

The National Bureau of Standards developed the Integrated Manufacturing Data Administration System (IMDAS) to support its Automated Manufacturing Research Facility (AMRF). AMRF represents an experimental facility for the research of small batch manufacturing. This environment is representative of other FMS and CIM environments in that it contains a heterogenous distributed database environment for managing manufacturing design, planning, and control databases.

IMDAS is characterized by:

- a common interface to user programs, and
- a common interface to underlying databases. [Ref. 18:p. 46]

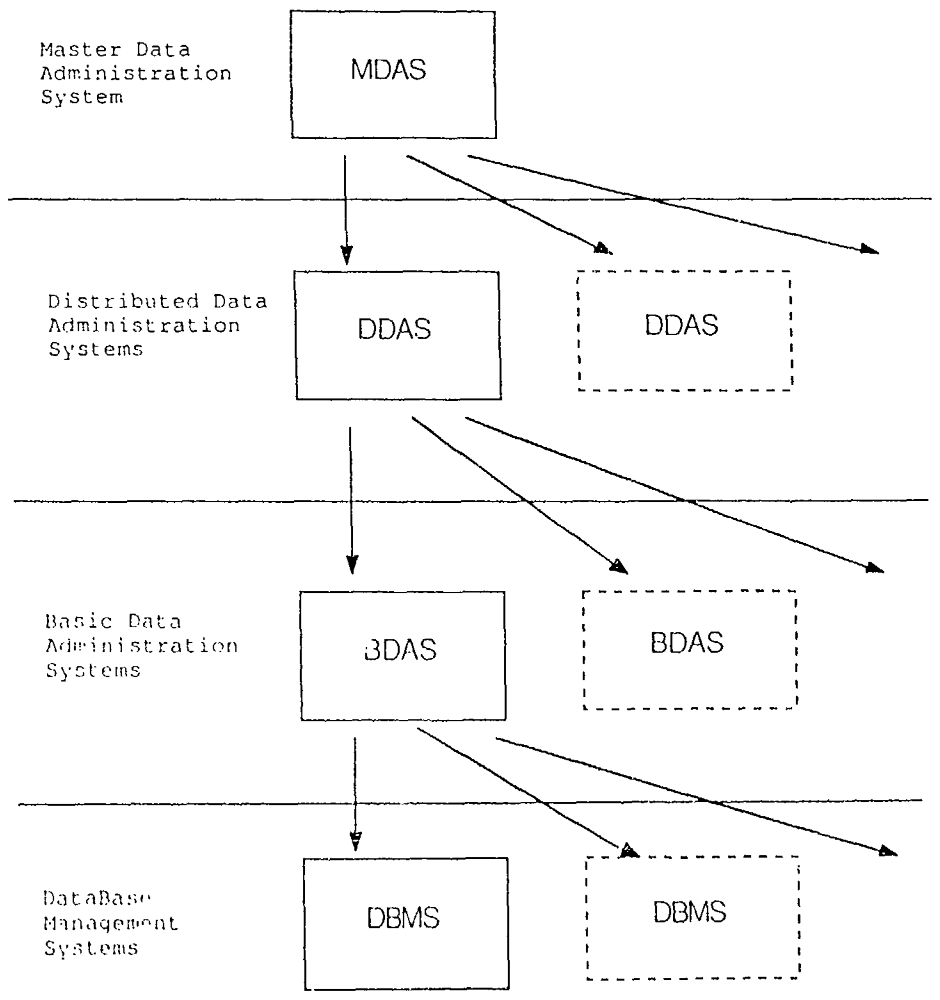
These interfaces provide an environment in which programmers need not concern themselves with the physical location of data, but rather are given the logical view of a single common database. Communication with IMDAS occurs through use of the ANSI-standard Structured Query Language (SQL) which has been modified to allow programs to specify files or buffers as sources and sinks of data.

IMDAS is structured in a four-level hierarchy, as depicted in Figure 8. The higher the level in this tree-like structure, the more data is administered. Since the lowest level is not an actual level per se, but depicts instead the local DBMS's throughout the system, the discussion of IMDAS begins with the Basic Data Administration System (BDAS).

a. Basic Data Administration System

Providing a common interface between user programs and databases is the premise upon which IMDAS is built. It is actually at this level, the BDAS, that this functi takes place.

The Interprocess Communications function provides the mechanism through which various processes can communicate with each other. Under the shared memory approach employed by IMDAS, an originating process stores information of a particular kind or for a particular recipient into a designated area of the shared memory, and a retrieving process interested in a particular information set fetches it from the pre-assigned area. [Ref 20: p. 20] This approach facilitates real-time data acquisition.


Master Data Administration System
MDAS
Distributed Data Administration Systems
DDAS
DDAS
Basic Data Administration Systems
BDAS
BDAS
DataBase Management Systems
DBMS
DBMS

Source: [Ref. 18]

Figure 8. The IMDAS Hierarchy

Interface between the standard IMDAS data form and the underlying DBMS is provided by the command translator/data translator (CT/DT) process. Data is translated from the representation provided by the local (source) DBMS to the IMDAS or common representation, and then from the common format to that of the destination DBMS. The global data manipulation language mentioned previously is translated into the "language" understood by the local DBMS or file manager.

The BDAS, in conjuction with the local DBMS provides the actual access to and manipulation of the local databases [Ref. 20:p. 22]. This function provides assistance in the DA functions of consistency, concurrency and recovery by providing the capability to read and write to the local files, as well as locking and unlocking them to prevent access when necessary.

As depicted in Figure 8, there are multiple BDAS peers in the hierarchy. The Basic Service Executive provides a c mmon interface between these BDAS's and the Distributed Data Administration System above them.

b. Distributed Data Administration System

The Distributed Data Administration System (DDAS) provides the interface between the BDAS(s) below it and the Master Data Administration System to be discussed next. There are multiple DDAS's, each responsible for interfacing with a particular group of component systems.

Each DDAS contains a data directory for the local database assigned to it as well as directory information from its subordinate BDAS's. This fragmentation directory provides a conceptual view representing the domain of that DDAS. The following information is included:

- the schema and mapping information needed for the global conceptual and fragmented views of data residing on subordinate BDAS's;
- the mapping information for the external and conceptual views of the data referenced by the control processes which that DDAS supports;
- the security constraints associated with the views of the data referenced by the control processes which that DDAS supports;
- the integrity constraints associated with data in the subordinate BDAS's;
- the data delivery information required by the network interprocess communication functions to construct delivery paths; and,
- the information representing the capability of each subordinate DBMS or Command Translator. [Ref. 20:p. 30]

When data are retrieved from multiple sources, or limited DBMS services are provided to one of the local databases, a Data Assembly Service (DAS) assembles data using joins, unions, and intersection record merging. Note that the IRDS can be the repository for the fragmentation directory.

A transaction manager (TM) schedules a collection of tasks (transaction) for execution based upon other activity in the system. Where necessary, it distributes different tasks among several BDAS's for execution. The two-phase locking approach used for concurrency control requires that any transactions requiring access to data that is currently being referenced or modified must wait until that process is concluded before proceeding. Finally, a recovery function brings the database back to a consistent state following a failure or integrity violation. Those tasks that cannot be accomplished by the DDAS are passed onto the Master Data Administration System. Note that the TM is essentially the ROM in the RAMP world.

c. Master Data Administration System

At the highest level of the hierarchy is the Master Data Administration System (MDAS) whose function is to coordinate the activities of the DDAS's. These activities include:

- managing the master data directory,
- resolving concurrency problems,
- directing initialization.
- directing integration and recovery.

Its construction is virtually identical to that of the DDAS's. Instead of all of the metadata provided by the directory function of the DDAS, however, the directory function of the MDAS describes the DDAS's. Its TM coordinates execution of tasks among various DDAS's.

As a final note, each DDAS contains the needed code to perform the task of MDAS. The system manager can therefore designate an MDAS in the event of a system crash or failure.

The RAMP architecture uses a similar hierarchical structure. The BDAS is basically equivalent to running an application program at the workstation level. While the cells in RAMP perform limited transaction management functions, they do control various workstations in a manner akin to the DDAS controlling the BDASs. By combining the ROM with the IRDS, along with the transaction manager found in RAMP, overall control of the cells is managed similar to the manner in which MDAS controls the various DDASs. The major deviation from this hierarchy lies in the fact that individual cells cannot be designated as the ROM like any DDAS can be designated an MDAS.

E. SUMMARY

This chapter began by providing a working definition for distributed computer processing. It then discussed why such an environment is necessary to support computer integrated manufacturing and flexible manufacturing systems. After discussing some of the data administration problems inherent in such an environment, the pivotal role of an IRDS in facilitating data administration in a heterogeneous distributed database environment was discussed. Finally, the chapter concluded with an overview of a data administration architecture under development by the Bureau of Standards to manage this environment and suggessted that the

IRDS is a tool which could be used to implement the IMDAS approach.

RAMP is based upon the IMDAS architecture and some of the generic data administration issues discussed [illegible] this chapter have been considered in its design. In Chapter IV we will review some of the additional RAMP-specific data administration problems which need to be resolved.

IV. DATABASE ADMINISTRATION ISSUES IN RAMP

A. INTRODUCTION

The RAMP SMP Manufacturing System is designed to satisfy the need within the Navy to reduce the lead time and costs associated with procuring spare and replacement parts. Flexible Manufacturing can provide the capability to produce needed parts in small quantities and to produce them quickly. Flexible Manufacturing Systems (FMS) are flexible because they can produce a wide variety of parts, and because new technology and products can be added to the system without significant reconfiguration of hardware and software [Ref. 6:p. 343]. An FMS integrates heterogeneous equipment, hardware, software, and data systems. It is this integration of the various technologies necessary to a manufacturing environment which is the challenge in designing an FMS.

Integrated and flexible systems such as RAMP require distribution of processing and sharing of data. Data administration in a distributed environment is complex and distribution which includes different hardware, software and database structures, is even more so.

The previous chapter discussed distributed processing and its inherent data administration problems. Now we examine more closely the structure of an Information

Resource Dictionary System (IRDS) and suggest ways in which an IRDS can be tailored to the RAMP manufacturing environment. Data administration problems introduced in Chapter III are also discussed in terms of the RAMP SMP Manufacturing System environment.

B. NEED FOR AN INFORMATION RESOURCE DICTIONARY SYSTEM

There is a need for extensive data sharing in the RAMP SMP System. The entire process of manufacturing a part or parts from the receipt of an order to the shipment of the part is heavily dependent upon ensuring that the data, such as Electronic Part Technical Data and Process Plans, are available for each application. Much of the data maintained in the data stores are used by more than one application or program. However, in Flexible Manufacturing Systems such as RAMP, each application may require the data to be organized or structured in different formats [Ref. 18:p. 2].

1. Data Organization in RAMP

The Check_for_Capacity_Problem and the Determine_Capacity_Availability modules within the Capacity_Requirements Planning program of the Production and Inventory Control component provide an example of data organization and use within the RAMP SMP System (Figure 9). The Check_for_Capacity_Problem module is initiated by an activity notification from the Estimate_Processing_Times module of Manufacturing Engineering component. This module accesses the Process_Plan data store to obtain Process_

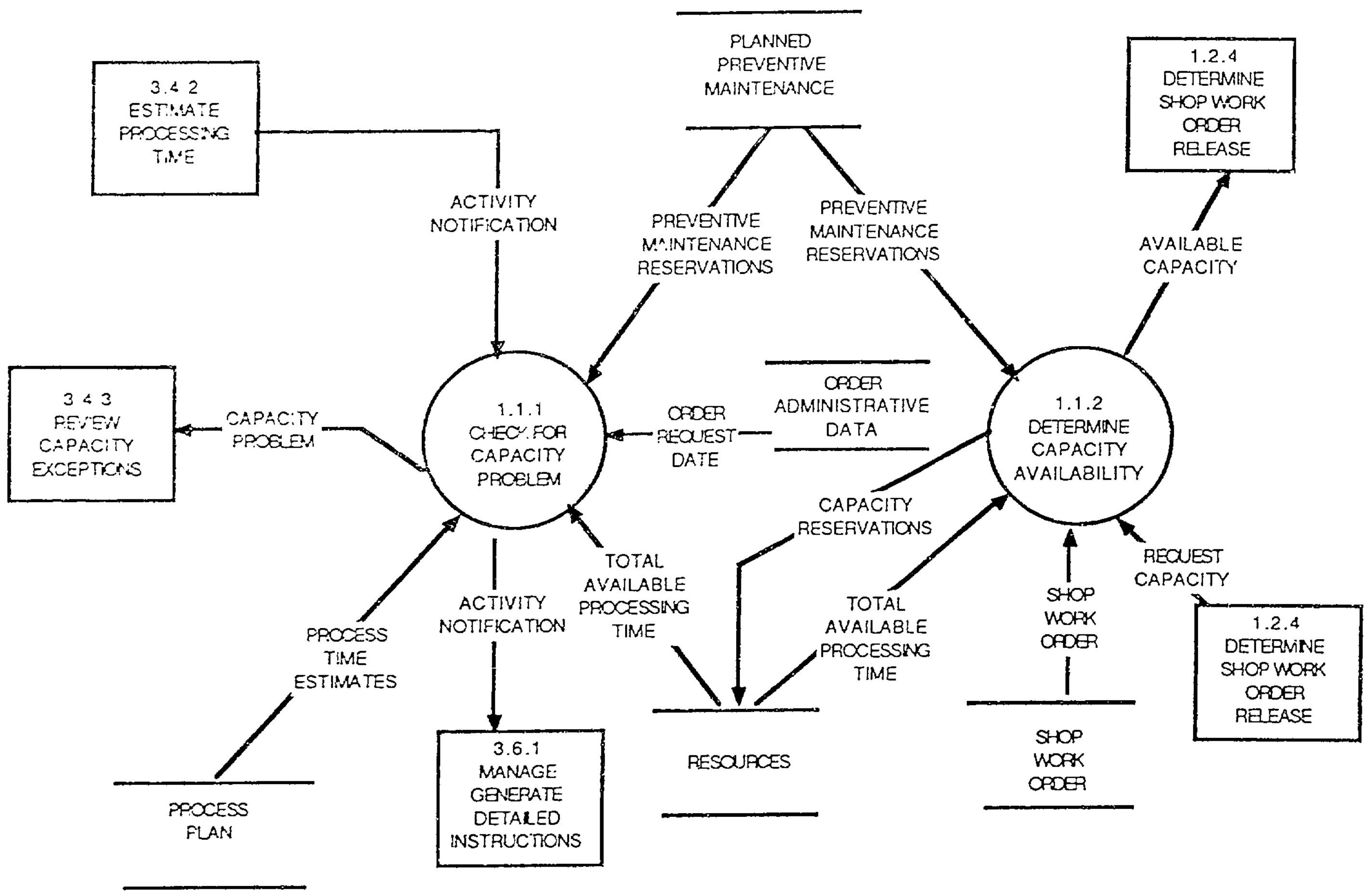

3.4.2 ESTIMATE PROCESSING TIME
ACTIVITY NOTIFICATION
PLANNED PREVENTIVE MAINTENANCE
PREVENTIVE MAINTENANCE RESERVATIONS
PREVENTIVE MAINTENANCE RESERVATIONS
1.2.4 DETERMINE SHOP WORK ORDER RELEASE
AVAILABLE CAPACITY
3.4.3 REVIEW CAPACITY EXCEPTIONS
CAPACITY PROBLEM
1.1.1 CHECK FOR CAPACITY PROBLEM
ORDER REQUEST DATE
ORDER ADMINISTRATIVE DATA
1.1.2 DETERMINE CAPACITY AVAILABILITY
CAPACITY RESERVATIONS
REQUEST CAPACITY
TOTAL AVAILABLE PROCESSING TIME
ACTIVITY NOTIFICATION
PROCESS TIME ESTIMATES
TOTAL AVAILABLE PROCESSING TIME
SHOP WORK ORDER
1.2.4 DETERMINE SHOP WORK ORDER RELEASE
3.6.1 MANAGE GENERATE DETAILED INSTRUCTIONS
RESOURCES
SHOP WORK ORDER
PROCESS PLAN

Figure 9. Data Organization for Capacity Requirements Planning

Time_Estimates. An Order_Request_Date is obtained from the Order_Administrative data store. This date is used to establish the time frame in which the part must be produced. The Resources data store is accessed to obtain the Total_Available_Processing_Time within the established time frame. Preventive_Maintenance_Reservations (scheduled down time) are obtained from the Planned_Preventive_Maintenance data store. All of the data (Process_Time_Estimates, Order_Request_Date, Total_Available_Processing_Time, Preventive_Maintenance_Reservations) are used by the Check_for_Capacity_Problem module to determine if the required amount of processing time will exceed the net available time. If the required time does exceed available capacity, a capacity problem message is sent to the Review_Capacity_Exceptions module of the Manufacturing Engineering component. [Ref. 2[illegible]:p. 1]

The Determine_Capacity_Availability module is initiated by a "request capacity" notice from the Determine_Shop_Work_Order_Release module. The Determine_Capacity_Availability module also accesses the Resources data store to obtain the Total_Available_Processing_Time and the Planned_Preventive_Maintenance data store to obtain Preventive_Maintenance_Reservations. Additionally, a committed completion date is obtained from the Shop_Work_Order data store. These data (Total_Available_Processing_Time, Preventive_Maintenance_Reservations, Committed_Completion_Date) are used to determine available capacity. If there is sufficient capacity available for the work order, a Capacity_Reservation is applied to the Resources data store and an "available capacity" message is sent to Determine_Shop_Work_Order_Release. [Ref. 21:p. 2]

As depicted in Figure 9, both of these modules require some of the same data. Each module also requires additional data from different data stores in order to yield the information necessary for each to perform its function.

2. Data Management

A relational data model is appropriate to capture the data needed by the systems, applications and programs. As the raw data are used, shared and manipulated by different applications for different purposes, the need for data integrity and consistency becomes paramount. A Data Base Management System (DBMS) to manage and control usage of, and access to, the data is necessary to ensure integrity and consistency. A Data Dictionary/Directory System (DD/DS), as an integral part of the DBMS, similarly manages and controls the metadata, or data about the data [Ref. 17:p. 11].

By forcing all applications to depend on the DD/DS for metadata, the DD/DS controls the component programs' and processes' access to data and further ensures data integrity and consistency [Ref. 17:p. 17]. Such a DD/DS is considered to be active.

In addition to providing consistent metadata, an active DD/DS provides other benefits as well. An active DD/DS controls the usage of the metadata and provides for greater data security. Data used by more than one component system or program need to be defined only once. Changes to the metadata are controlled and more easily propagated throughout a distributed system. Generating metadata for the user programs allows for greater data independence. [Ref. 17:p. 119] Documentation of the database is also more accurate and timely [Ref. 17:p. 144].

The RAMP SMP environment is comprised of several different software systems and hardware configurations necessary to achieve efficient equipment operation on the factory floor. The various automated systems and equipment in this heterogeneous environment are distributed over an Open Systems Interconnection (OSI) communication network [Ref. 7:p. 35]. Multiple databases are accessed by multiple functional components, programs or modules. Because there exist multiple databases and DBMSs in the RAMP environment, a system-wide DD/DS is required which is independent of any specific DBMS. This DD/DS, in conjunction with the ROM, form the equivalent of the MDAS in the NBS model.

To simplify the multiple database/multiple DBMS environment, the RAMP SMP System takes advantage of commercial off-the-shelf (COTS) DBMSs to manage local databases. Data needed by a functional component application must be converted into the local format when down loaded from the common database. [Ref. 9:p. 3.3.1.3-R9] Conversely, data required by more than one cell or workstation will be up loaded to the common database upon the completion of an application. This common data will necessarily undergo a translation or conversion from the local database format to the common database format.

A relational data model of this integrated database environment is appropriate. However, a typical DD/DS provided by relational systems is limited in the capabilities it can provide. An Information Resource Dictionary System (IRDS) is an expanded DD/DS, which is much broader in scope and capable of capturing more data to model precisely the RAMP SMP system. While a typical DD/DS is concerned only with the data resource, an IRDS contains metadata about other resources such as processes, programs, hardware, and users of the system, as well. In addition, the IRDS captures how all these resources are interrelated. A relational DBMS (RDBMS), through its DD/DS, may contain data about a parts file such as part number, vendor and price. An IRDS may contain not only this metadata but also which programs, such as an inventory_control_program, process this file and which users run the inventory program. Relationships are defined as well as data elements. [Ref. 15:pp. 48-49]

C. INFORMATION RESOURCE DICTIONARY SYSTEM OVERVIEW

The Federal Information Processing Standard (FIPS) specifies an IRDS which provides the previously mentioned benefits of a DD/DS plus the capability to tailor the dictionary to a specific environment [Ref. 15:p. 48].

1. Core System-Standard Schema

The FIPS IRDS is comprised of a Core System-Standard Schema. This core IRDS consists of the Information Resource Dictionary (IRD) and the IRD Schema. The IRD Schema describes and controls the IRD which in turn describes and controls the actual data. [Ref. 22:p. 115] Based on the entity-relationship model, the IRDS is strongly typed so that each entity, relationship and attribute in the IRD is an instance of an entity-type, relationship-type or attribute-type in the IRD Schema, respectively. Relationships are directed and binary, and entities may be related to themselves. Both entities and relationships may have associated attributes. [Ref. 22:p. 114]

2. Information Resource Dictionary (IRD) Schema

The IRD Schema is defined in terms of the entity-relationship model and consists of entity-types, relationship-types and attribute-types which describe the entity, relationship and attribute instances of the IRD. As shown in Table 7, there are 12 entity-types in the core system-standard schema. Eight of these are Data entity-types: File, Record, Element, Document, Bit-String,

TABLE 7

CORE SYSTEM-STANDARD SCHEMA TYPES

Entity-types

SYSTEM	FILE	BIT-STRING
PROGRAM	RECORD	CHARACTER-STRING
MODULE	ELEMENT	FIXED-POINT
USER	DOCUMENT	FLOAT

Attribute-types

Entity-related:

ACCESS-NAME	DURATION-VALUE
ADDED-BY	HIGH-OF-RANGE
ALLOWABLE-VALUE	LAST-MODIFICATION-DATE
ALTERNATE-NAME	LAST-MODIFIED-BY
CLASSIFICATION	LOCATION
CODE-LIST-LOCATION	LOW-OF-RANGE
COMMENTS	NUMBER-OF-LINES-OF-CODE
DATA-CLASS	NUMBER-OF-MODIFICATIONS
DATE-ADDED	NUMBER-OF-RECORDS
DESCRIPTION	RECORD-CATEGORY
DESCRIPTIVE-NAME	SECURITY
DOCUMENT-CATEGORY	SYSTEM
DURATION-TYPE	

Relationship-related:

ACCESS-METHOD	FREQUENCY
RELATIVE-POSITION	

Relationship-types

CONTAINS	GOES-TO
PROCESSES	CALLS
RESPONSIBLE-FOR	DERIVED-FROM
RUNS	REPRESENTED-AS

Source: [Ref. 15:p. 50]

Character-String, Fixed-Point, and Float. System, Program, and Module are Process entity-types and User is an External entity-type. [Ref.15:pp. 50-51]

The attribute-types are associated either with entity-types or with relationship-types. Some attribute-types are common to all entity-types, such as Description and Comments. Ot'❋r attribute-types, such as Number-of-Lines-of-Code and Number-of-Records, are associated with only one or a few entity-types. Audit trail information is provided by attribute-types such as Date-Added and Last-Modified-By. The attribute-types Access-Method, Relative-Position, and Frequency are associated with relationship-types. [Ref. 15:p. 51]

The core system-standard relationship-types describe the associations between entity-types. ɔr example, Program-Contains-Module and Document-Derived-from-File are relationship-types. Limiting which entity-types may participate in which relationship-types helps to maintain integrity within the IRDS. For example, because relationships are directed, the relationship File-Contains-Record is valid while Record-Contains-File is not. Allowable relationships, in the format Relationship(Entitytype, Entitytype), are listed in Table 8. [Ref. 15:p. 51]

3. <u>IRDS Facilities</u>

An Extensibility facility allows additional schema descriptors to be added to the core system-standard schema

TABLE 8

IRDS RELATIONSHIPS

CONTAINS(system,system)
CONTAINS(system,program)
CONTAINS(system,module)
CONTAINS(program,program)
CONTAINS(program,module)
CONTAINS(module,module)
CONTAINS(file,file)
CONTAINS(file,document)
CONTAINS(file,record)
CONTAINS(file,element)
CONTAINS(document,document)
CONTAINS(document,record)
CONTAINS(document,element)
CONTAINS(record,record)
CONTAINS(record,element)
CONTAINS(element,element)

PROCESSES(system, file)
PROCESSES(system,document)
PROCESSES(system,record)
PROCESSES(system,element)
PROCESSES(program,file)
PROCESSES(program,document)
PROCESSES(program,record)
PROCESSES(program,element)
PROCESSES(module,file)
PROCESSES(module,document)
PROCESSES(module,record)
PROCESSES(module,element)
PROCESSES(user,file)
PROCESSES(user,document)
PROCESSES(user,record)
PROCESSES(user,element)

RESP_FOR(user,file)
RESP_FOR(user,document)
RESP_FOR(user,record)
RESP_FOR(user,element)
RESP_FOR(user,system)
RESP_FOR(user,program)
RESP_FOR(user,module)

RUNS(user,system)
RUNS(user,program)
RUNS(user,module)

DERIVED_FROM(document,file)
DERIVED_FROM(document,document)
DERIVED_FROM(document,record)
DERIVED_FROM(element,file)
DERIVED_FROM(element,document)
DERIVED_FROM(element,record)
DERIVED_FROM(element,element)
DERIVED_FROM(file,document)
DERIVED_FROM(file,file)
DERIVED_FROM(record,document)
DERIVED_FROM(record,file)
DERIVED_FROM(record,record)

CALLS(program,program)
CALLS(program,module)
CALLS(module,module)

GOES_TO(system,system)
GOES_TO(program,program)
GOES_TO(module,module)

Source: [Ref. 15:p. 51]

as needed by the user or dictated by the environment [Ref. 15:p. 50].

The Quality-Indicator facility of the IRDS provides a method of defining quality indicators and assigning them to entities. Quality indicators are metaentities in the IRD Schema useful for documentation, such as documenting the level of standardization of element entities. [Ref. 22:p. 120]

Logical subsets of the IRD are defined as Views. A View is a set of entities, along with their associated attributes, the set of relationships that exist between the entities, and the set of operations that may be performed in that view. [Ref. 22:p. 120]

Core security is provided by a Dictionary-User entity which has attributes to define level of access. Additionally, the relationship Dictionary-User-Has-View determines the views, or logical subsets of the IRD, which may be accessed by the user. [Ref. 22:p. 120]

Two user interfaces are specified by the IRDS, a menu driven panel interface and a command language interface. Either one or both may be included in an IRDS implementation. [Ref. 15:p. 50]

In the following section an expanded IRD schema for the RAMP database architecture is proposed. This discussion will focus only on the core system-standard schema and will not consider additional features.

D. EXPANDED IRD SCHEMA FOR RAMP

A Flexible Manufacturing System such as RAMP includes some entity-types and relationship-types not described in the core system-standard schema discussed in the previous section. This section attempts to identify those characteristics which are unique to the RAMP SMP Manufacturing System and to define schema descriptors for these characteristics which could be added to the core IRD Schema.

1. Entity-types

Most, if not all, of the entity-types comprising the core system-standard schema are useful in implementing an IRDS in the RAMP database architecture. Capacity_ Requirements_Planning and Production_Control are two instances of the entity-type PROGRAM. The program modules Check_for_Capacity_Problem and Determine_Capacity_ Availability are instances of the entity-type MODULE. Data stores such as Tool_Inventory, Shop_Work_Order, Order_ Status, and Planned_Preventive_Maintenance are all instances of the entity-type FILE.

Levels of the RAMP control hierarchy (cell, workstation, equipment) can be designated as entity-types in an expanded IRD Schema. Instances of the entity-type CELL are: Production_and_Inventory_Control_Cell, Manufacturing_ Engineering_Cell, Manufacturing_Cell, Quality_Cell, Information_Management_Cell, and Communications_Cell.

Material_Preparation, Horizontal_Machining_Center, Turning_Center, and Shipping_Workstations are a few examples of instances of the entity-type WORKSTATION. Lathe, Drill_ Grinder and Band_Saw are instances of the entity-type EQUIPMENT.

At the System level of the RAMP hierarchy is the RAMP Order Manager (ROM) [Ref. 9:p. 3.1-7L]. Interfaces to ROM are provided by Command/Status Services (Figure 10). The relationship between ROM and the IRDS for the common database is active. Individual applications at the local level are dependent upon the IRDS which manages the metadata of the common database. It is the ROM which provides local applications with access to the common data by receiving status reports, such as "down load complete" or "execution complete," and issuing commands such as "execute applica-ion" or "up load data." COMMAND and STATUS can be defined as entity-types. Down_Load_Complete and Execution_Complete are instances of the entity-type STATUS. Execute_Applica-tion and Up_Load_Data are instances of the entity-type COMMAND.

Other entity-types to be defined in an expanded IRD are the entity-types PROCESS and APPLICATION. As discussed in Chapter II, the ROM coordinates and manages the "process" or sequence of applications necessary to complete a function or request. Two instances of the entity-type PROCESS are Order_Process and Order_Inquiry_Process. Each commercial

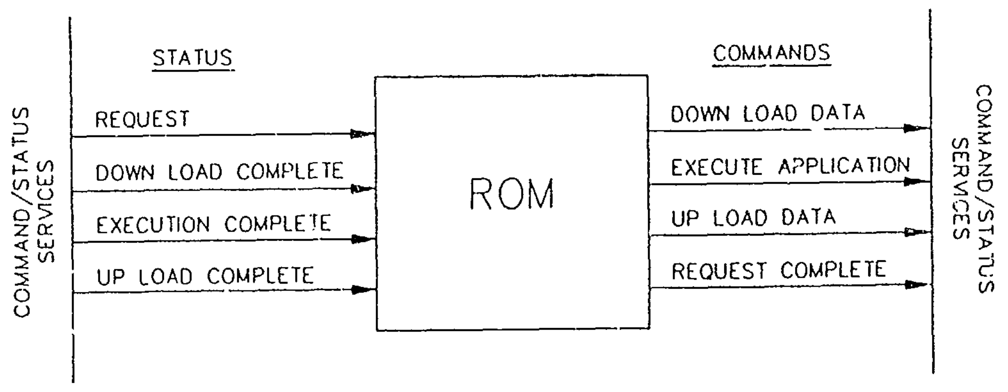


Figure 10. ROM Command/Status Services
[Ref. 9:p. 3.3.1-L4]

off-the-shelf application package is defined as an instance of the entity-type APPLICATION. Table 9 lists the entity-types of this expanded IRD Schema.

2. Relationship-types

The core relationship-type CONTAINS is a necessary schema descriptor in the RAMP database architecture. Entity-types of the expanded IRD participate in the relationship-type CONTAINS as follows: CELL-CONTAINS-WORKSTATION, WORKSTATION-CONTAINS-EQUIPMENT and PROCESS-CONTAINS-APPLICATION.

The core relationship-type PROCESSES is not necessary to an IRD Schema in the RAMP database architecture. SYSTEM-PROCESSES-FILE is a typical PROCESSES relationship-type in the core system-standard schema [Ref. 22:p. 116]. In the RAMP SMP Manufacturing System, files are not "processed" as such, but rather data is obtained from

TABLE 9

EXPANDED IRD SCHEMA TYPES

Entity-types

SYSTEM	PROCESS	CELL	FILE
PROGRAM	APPLICATION	WORKSTATION	RECORD
MODULE	STATUS	EQUIPMENT	ELEMENT
USER	COMMAND		DOCUMENT

BIT-STRING
CHARACTER-STRING
FIXED-POINT
FLOAT

Relationship-types

CONTAINS	INITIATES
ACCESSES	CALLS
RESPONSIBLE-FOR	DERIVED-FROM
RUNS	REPRESENTED-AS

the file and used by an application to perform a function or manufacturing process. To capture this relationship, an additional relationship-type to be defined in an expanded IRD is ACCESSES. CELL-ACCESSES-FILE and APPLICATION-ACCESSES-FILE are relationship-types which define associations between the entity-types CELL, APPLICATION and FILE.

INITIATES is another relationship-type to be defined in the expanded IRD Schema. STATUS-INITIATES-COMMAND defines the relationship between the entity-types STATUS and COMMAND.

The relationship-types of the expanded IRD are shown in Table 9. Allowable relationships between the entity-types of the expanded IRD are listed in Table 10.

3. Flexibility

The RAMP SMP Manufacturing System is designed to be modular to provide for maximum flexibility. The system must allow for equipment and application programs to be added and deleted without a major disruption in the production of parts. The implementation of RAMP capabilities should be in stages. Initial RAMP capabilities should include both manual and automated processes. Some manual processes will become automated over time. The modularity of the system will allow for newly automated processes to be easily added to the system. An IRDS is able to capture the architecture of the system as it exists. As new capabilities and components are added to the system, they also are easily added to the IRDS. Additional schema descriptors can also be defined as necessary

Storing configuration tables in the dictionary provides a way for ROM to accurately direct and control the processes and applications being executed at the equipment and workstation level. The ROM is event-driven with status reports indicating the next action to be taken. The relationship-types STATUS-INITIATES-COMMAND and COMMAND-INITIATES-APPLICATION capture this information in the dictionary. If a new process is added to the capabilities

TABLE 10

EXPANDED IRDS RELATIONSHIPS

CONTAINS(cell,workstation)
CONTAINS(workstation,equipment)
CONTAINS(process,application)
CONTAINS(system,system)
CONTAINS(system,program)
CONTAINS(system,module)
CONTAINS(program,program)
CONTAINS(program,module)
CONTAINS(document,document)
CONTAINS(document,element)
CONTAINS(file,document)
CONTAINS(file,record)
CONTAINS(file,element)
CONTAINS(file,file)
CONTAINS(document,record)
CONTAINS(record,record)
CONTAINS(record,element)
CONTAINS(element,element)
CONTAINS(module,module)

ACCESSES(cell,file)
ACCESSES(cell,record)
ACCESSES(cell,element)
ACCESSES(equipment,file)
ACCESSES(equipment,record)
ACCESSES(equipment,element)
ACCESSES(module,file)
ACCESSES(module,element)
ACCESSES(module,record)
ACCESSES(application,file)
ACCESSES(application,element)
ACCESSES(workstation,file)
ACCESSES(workstation,record)
ACCESSES(workstation,element)
ACCESSES(program,file)
ACCESSES(program,record)
ACCESSES(program,element)
ACCESSES(process,file)
ACCESSES(process,record)
ACCESSES(process,element)
ACCESSES(application,record)

INITIATES(status,command)
INITIATES(command,application)

RESP_FOR(user,file)
RESP_FOR(user,document)
RESP_FOR(user,record)
RESP_FOR(user,element)
RESP_FOR(user,system)
RESP_FOR(user,program)
RESP_FOR(user,module)

RUNS(user,system)
RUNS(user,program)
RUNS(user,module)

DERIVED_FROM(document,file)
DERIVED_FROM(document,document)
DERIVED_FROM(document,record)
DERIVED_FROM(element,file)
DERIVED_FROM(element,document)
DERIVED_FROM(element,record)
DERIVED_FROM(element,element)
DERIVED_FROM(file,document)
DERIVED_FROM(file,file)
DERIVED_FROM(record,document)
DERIVED_FROM(record,file)
DERIVED_FROM(record,record)

CALLS(program,program)
CALLS(program,module)
CALLS(module,module)

GOES_TO(system,system)
GOES_TO(program,program)
GOES_TO(module,module)

of the RAMP SMP Manufacturing System, the configuration table is modified to reflect this new process or application.

E. DATA ADMINISTRATION

The discussion in Chapter II included data administration policies necessary in a computer integrated manufacturing environment to provide for database integrity, security, concurrency, transaction granularity, back-up and recovery, and information resource management. The previous section addressed the issue of information resource management and suggested an Expanded IRD Schema for the RAMP architecture. The discussion which follows examines the remaining data administration issues in the context of the specific RAMP SMP Manufacturing System.

1. Database Integrity

Although it is impossible to ensure that only correct data are entered into the common databases, the IRDS can validate the range of an entry into the database. The attribute-types LOW-OF-RANGE and HIGH-OF-RANGE can be associated with the ELEMENT entity-type. While this method will ensure that the entry is in the proper range, it cannot ensure a correct value.

As discussed earlier, integrity constraints are also enforced by the IRDS by limiting which entity-types can participate in which relation-types. ELEMENT-CONTAINS-RECORD is not a valid relationship, for example. Dolk [Ref.

15] shows how prestored SQL commands can be used to check that only proper relationships have been entered in the IRD.

Including the data formats of both the local databases and the common databases in the IRDS improves data integrity by enforcing the proper format structure. Data being up loaded to the common database must be validated as being in the common database format. When ROM, through command/status services, issues the command to down load data, the IRDS plays an active role by locating the data in the common database and facilitating the conversion by also defining the local data format.

2. Database Security

The common database is accessed only by the RAMP Order Manager (ROM) through the DBMS of the Information Management function. By including in the IRDS what data are needed by which applications, the ROM can ensure that only necessary data are down loaded to the functional components. This information is captured in the expanded IRD relationship-types APPLICATION-ACCESSES-FILE and APPLICATION-ACCESSES-RECORD. Each process, or sequence of applications, must be defined in terms of the common data needed by these application programs. This is similar to access and capability lists. However, only the ROM issues the commands to down load data.

In the RAMP SMP database architecture, local databases are accessed only by the applications local to the functional component at that particular node. These applications are unaware of data that may exist at any other node. Because of this partitioning of the data, each node must be responsible for providing some degree of security for the local data.

3. Database Update and Concurrency Control

The RAMP database architecture is designed to eliminate many concurrency issues. The RAMP Order Manager (ROM) manages all applications and processes all requests. The ROM is event driven and sequential in nature. Commands are issued by the ROM upon receipt of status data. Data shared by more than one functional component reside in a common database and are down loaded to a local database when necessary to an application. Due to the sequential nature of the applications, the next application is not initiated until the previous application has completed and the data, which may have been modified, are up loaded to the common database. This method does not allow two applications to simultaneously update the same data. Neither can one application read data while another is updating the same data. [Ref. 9:p. 3.3.1.2-R10-R11]

At any point in time, common data residing at local databases will not necessarily be consistent. However, under normal operating conditions it is irrelevant whether the correct data reside at the local level at any particular time because an application receives the correct data from the common database immediately before that application executes.

4. <u>Transaction Granularity</u>

Logical transactions in the RAMP SMP Manufacturing System must be defined to ensure that all databases can be recovered in the event of a failure in software, hardware or equipment. The determinations of how many actions should be included in a logical transaction is critical. For example, suppose a transaction includes an entire process, or sequence of applications and upon completion of the third application, a data up load fails. The local database may be correct and current while the common database, upon which all applications depend, is not. If the process is to be restarted, the common database may be recovered to undo all actions taken before the process began. But what program has responsibility for recovering the local databases which have been changed by this partially executed process? The common data residing at the local level will be updated the next time data are down loaded, but any local data which were changed as a result of the completed applications may now be incorrect if the process must be restarted.

At the local level transactions cannot by definition encompass more than a single application. The transaction begins when the Process Manager component of ROM requests the Application Manager to invoke an application and ends when the ROM Manager sends the status message indicating that the data up load is complete. At that point, data has been physically committed to the local database. However this commitment may not (probably will not) coincide with the transaction boundary of the calling process. Thus local databases may be committed while the common global database is not. This can cause serious inconsistencies if a system failure occurs.

The inconsistencies which may occur between the local databases and the common databases present another problem. If a part has completed operations at the horizontal machining center and is now ready to undergo deburring when a data up load fails, what happens to the part? How much manual intervention is required to restart the physical process of deburring? Where is the manufacturing process restarted physically and how are the common database and local databases recovered through rollback/ rollforward to reflect that precise point?

If a failure occurs at the equipment level, it will be necessary for ROM to reinitiate a process or application. Here again the question arises of restarting a physical process and ensuring that the database is rolled back to reflect the correct data.

Another more dire possibility is that the ROM could fail. In this case, it is necessary to know which applications have been completed. Without access to the application checkpoint file it may not be possible to determine this. Which local databases reflect the current state and which do not? Is it possible to rollback all local databases to the same point in time and could the common database be reconstructed and recovered?

Using a hybrid scheme of data placement by replicating some of the data in the RAMP architecture and partitioning the rest is a feasible alternative in a heterogeneous environment. It does, however, present the nontrivial problem of maintaining consistency among the databases should a failure occur.

5. Database Back-up and Recovery

As discussed in Chapter II, Before-and-After images of updates to the common database will be recorded in a recovery log. In the event of a program failure or system hardware or software failure, Roll Back will be initiated to recover the database. If the disk media fails, an archived database and the recovery log will be used to Roll Forward the database. [Ref. 8:p. 191]

Writing Before-and-After images to a recovery log is a relatively straightforward approach. The major difficulties are defining the transaction granularity and ensuring consistency among the local databases and the common databases, as already discussed, and the related issue of the amount of storage space required for the recovery log.

In addition to back-up and recovery procedures for the common databases, it is also necessary for each local database to provide for back-up and recovery of data which reside only at a functional component node. Since this is a function of the COTS which "owns" the local database, this may be problematic.

F. SUMMARY

Data administration in a distributed environment such as RAMP presents many difficulties. Many of these are problems inherent in distributed systems. The manufacturing environment presents another set of problems which must be addressed. In order to provide the most efficient operation on the factory floor, a variety of equipment, computer systems, hardware, and software must be integrated and controlled.

Data administration in a distributed environment where some of the data are replicated and the rest partitioned becomes quite complex. In RAMP, the RAMP Order Manager (ROM) controls the down loading of data from, and the up loading of data to, the common database. An Information Resource Dictionary System (IRDS) is suggested to describe, define and locate the data of the RAMP common database. The IRDS is expanded to include characteristics peculiar to the RAMP SMP Manufacturing System environment.

Database integrity, security, update, and concurrency control are all issues which must be addressed both at the common database level and at the local level. Defining the size of a transaction or logical unit of work is critical to ensuring adequate database back-up and recovery. Many questions need to be resolved. Procedures must be defined to ensure database recovery should a failure of equipment or software occur at any level within RAMP. In particular, the database must be recovered to coincide with the physical process of manufacturing parts.

V. SUMMARY, CONCLUSIONS AND RECOMMENDATIONS

A. SUMMARY

The RAMP prototype facility at Charleston Naval Shipyard (CNSY) will employ state of the art technologies to develop small mechanical parts for the United States Navy. As a heterogeneous flexible manufacturing system (FMS), it will have the ability to reconfigure its hardware and software without significant disruption to production requirements. RAMP is expected to provide a tremendous improvement over the current procurement processes by reducing lead time requirements up to 90%. If successful, it will be permanently installed at CNSY and other Naval facilities, with the potential for installation at additional private facilities.

Chapter II provided the reader with an introduction to RAMP. A brief overview of the the current procurement process was provided, followed by a detailed discussion of the RAMP functional components, and how RAMP is expected to improve the production process.

Chapter II continued with an overview of the data placement and management schemes in use at RAMP. While data placement was determined to be hybrid in nature, data management and control is centralized through the RAMP Order Manager (ROM). ROM's three functional components--ROM

Manager, Process Manager, and Application Manager--coordinate processing and data transfer requirements between the global and local databases.

Chapter III reviewed distributed processing concepts, and discussed why distribution is necessary in the CIM and FMS environments. The functions normally encountered in CIM environments, as well as those in use at RAMP were briefly reviewed. Data administration considerations were then discussed, with particular emphasis on the need for a information resource dictionary system (IRDS). Finally, the Integrated Manufacturing Database Administration System (IMDAS) developed by the National Bureau of Standards (NBS) to address some of the complexities encountered was discussed.

Chapter IV related the data administration considerations of Chapter III to the specific RAMP environment. While concurrency problems appear to be negligible in RAMP, the remaining issues of transaction granularity, security, backup and recovery and overall database integrity need to be considered. The chapter provided a detailed review of an expanded IRDS geared specifically to the RAMP environment to facilitate data administration. It is this discussion which forms the basis for our conclusions and recommendations which follow.

B. CONCLUSIONS AND RECOMMENDATIONS

The RAMP environment is a complex distributed system and requires extensive data sharing among the various functional components. Because of the multiplicity of databases and data base management systems (DBMSs) in RAMP, a system-wide data dictionary/directory system (DD/DS) is required. Since a typical DD/DS does not capture all of the relevant meta-data involved, it is recommended instead that an IRDS based upon the NBS architecture be employed.

IMDAS was developed as a Federal Information Processing Standard (FIPS) which could be easily and readily adapted to a specific FMS environment. The IRDS contains a core system-standard schema which can be adapted to the IMDAS architectureas well as RAMP. The extensibility feature also provides a mechanism by which the IRDS can be expanded to include any additional attributes, entities, and relationships necessary as the FMS evolves.

The FIPS IRDS forms an important basis for information management in RAMP. By including the additional entity-types encountered in RAMP--CELL, WORKSTATION, and EQUIPMENT--the IRDS can model the unique RAMP environment. While the relationship-type "PROCESSES" is not required by RAMP, the addition of "ACCESSES" should be included. The additional entity and relationship types discussed in Chapter IV can be readily included as well.

The centralized nature of the ROM provides an excellent mechanism by which to create an active IRDS facility. Individual applications at the local level are forced through the IRDS to enhance security and integrity constraints. As the system grows and changes in architecture, the IRDS can be easily expanded to include any additional schema descriptors which may become necessary.

The issue of defining logical transactions and database backup/recovery has not been resolved, however. The current ROM does not appear to provide adequate capabilities in these areas and may have to be expanded to accommodate these potential problems. The role of an IRDS in facilitating backup and recovery in a heterogeneous environment is a fruitful topic for future research.

LIST OF REFERENCES

1. Bryant, Mike, A Study of the Adequacy of Existing Navy Industrial Fund Cost Accounting Procedures for Flexible Manufacturing Systems, Master's Thesis, Naval Postgraduate School, Monterey, California, June 1988.

2. Bennet, Robert E., and others, Cost Accounting for Factory Automation, National Association of Accountants, 1987.

3. Naval Supply Systems Command, Department of the Navy, "RAMP Program Plan Summary," Washington, D.C., June 1986.

4. Houts, Robert E. Jr., "Logistics Research and Development--Near Term Plans," June 1986.

5. American Manufacturing Research Consortium, "Operational Concept Document for the RAMP Small Mechanical Parts (SMP) Manufacturing System," Small Mechanical Parts Type B Specifications for the RAMP/RTIF Program, RTIF Program Document Number OCR 003001-1, 27 January 1988.

6. Besant, C.B., and Lui, C.W.K., Computer-Aided Design and Manufacture, West Sussex, England: Ellis Horwood Limited, 1986.

7. American Manufacturing Research Consortium, "Prime Item Development Specification," Small Mechanical Parts Type B Specifications for the RAMP/RTIF Program, RTIF Program Document Number PIR 003001-1, 22 January 1988.

8. American Manufacturing Research Consortium, "Software Requirements Specification," Small Mechanical Parts Type B Specifications for the RAMP/RTIF Program, RTIF Program Document Number SRR 003001-1, 26 January 1988.

9. American Manufacturing Research Consortium, RAMP SMP Critical Design Review, 30 August-1 September 1988.

10. American Manufacturing Research Consortium, "Interface Requirements Specification," Small Mechanical Parts Type B Specifications for the RAMP/RTIF Program, RTIF Program Document Number IRR 003001-1, 26 January 1988.

11. Sloman, Morris, and Krenser, Jeff, Distributed Systems and Computer Networks, Prentice/Hall International, 1987.

12. Beeby, William D., "The Heart of Integration: A Sound Data Base," Computers and Manufacturing Productivity, Ronald K. Jurgen, ed., New York: IEEE Press, 1987.

13. Davis, W., Jones, A., and Ram, S., "Data Management Strategies for Computer Integrated Management Systems," unpublished paper, National Bureau of Standards.

14. Melkanoff, Michel A., "The CIMS Database: Goals, Problems, Case Studies and Proposed Approaches Outlined," Industrial Engineer, pp. 78-93, November 1984.

15. Dolk, Daniel R., and Kirsch, Robert A. II, "A Relational Information Resource Dictionary System," Communications of the ACM, Vol. 30, January 1987.

16. Bray, Olin, Distributed Database Management Systems, Lexington Books, 1982.

17. Leong-Hong, Belkis W., and Plagman, Bernard K., Data Dictionary/Directory Systems: Administration, Implementation and Usage, John Wiley & Sons, 1982.

18. Libes, Don and Barkmeyer, Edward "IMDAS--An Overview," undated article: Integrated Systems Group, National Bureau of Standards, Gaithersburg, Maryland.

19. Rauch-Hindin, Wendy, "True Distributed DBMSes Presage Big Dividends," Mini-Micro Systems, pp.65-73, May 1987.

20. U.S. Department of Commerce, National Bureau of Standards, An Architecture for Distributed Data Management in Computer Integrated Manufacturing, by Edward Barkmeyer and others, January 1986.

21. American Manufacturing Research Consortium, "Implementation Model," Small Mechanical Parts Type B Specifications for the RAMP/RTIF Program, RTIF Program Document Number IMR 001001-0, 19 November 1987.

22. Goldfine, Alan H., "The Information Resource Dictionary System," paper presented at the International Conference on Entity-Relationship Approach, 4th, Chicago, Illinois, 28-30 October 1985.

INITIAL DISTRIBUTION LIST

		No. Copies
1.	Defense Technical Information System Cameron Station Alexandria, Virginia 22304-6145	2
2.	Library, Code 0142 Naval Postgraduate School Monterey, California 93943-5002	2
3.	Director, Information Systems (OP-945) Office of the Chief of Naval Operations Navy Department Washington, D. C. 20350-2000	1
4.	Computer Technology Programs, Code 37 Naval Postgraduate School Monterey, California 93943-5000	1
5.	Naval Supply Systems Command (PML 5505M) Attn: Lorna B. Estep Department of the Navy Washington, D. C. 20376-5000	3
6.	Professor Daniel R. Dolk, Code 54DK Department of Administrative Sciences Naval Postgraduate School Monterey, California 93943-5000	3
7.	Professor Alan W. McMasters, Code 54MG Department of Administrative Sciences Naval Postgraduate School Monterey, California 93943-5000	1
8.	Professor Kenneth J. Euske, Code 54EE Department of Administrative Sciences Naval Postgraduate School Monterey, California 93943-5000	1
9.	Lieutenant Catherine T. Eads Navy Regional Data Automation Center 4400 Dauphine Street New Orleans, Louisiana 70145-7700	2
10.	Lieutenant Pamela A. Smith 10501 Curran Boulevard #4H New Orleans, Louisiana 70127	2